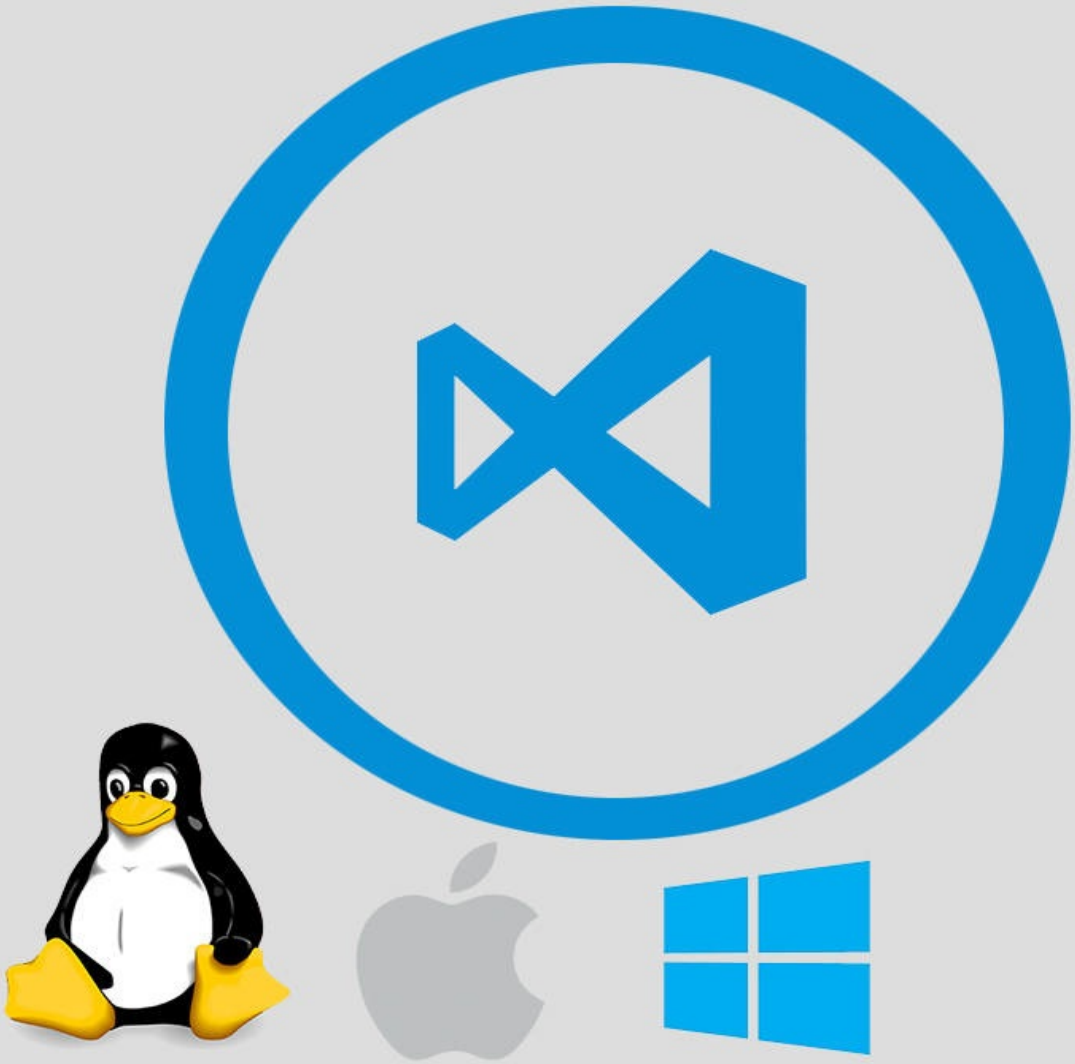
Getting Started with ASP.NET 5

for OS X, Linux, and Windows



Microsoft® Most Valuable Professional

Agus Kurniawan

Copyright

Getting Started with ASP.NET 5 for OS X, Linux, and Windows

Agus Kurniawan

1st Edition, 2015

Copyright © 2015 Agus Kurniawan

Table of Contents

Copyright

Preface

1. Preparing Development Environment

1.1 ASP.NET 5

1.2 System Requirements

1.3 Development Tools

2 Deploying ASP.NET 5 on OS X

2.1 Getting Started

2.2 Deploying ASP.NET 5

2.3 Testing

2.3.1 ASP.NET Console

2.3.2 ASP.NET 5

2.3.3 ASP.NET MVC

2.4 Development Tools

3 Deploying ASP.NET 5 on Linux

3.1 Getting Started

3.2 Deploying ASP.NET 5

3.2.1 Installing Mono

3.2.2 Installing libuv

3.2.3 Installing DNVM

3.2.4 Installing git

3.3 Testing

3.3.1 ASP.NET Console

3.3.2 ASP.NET 5

3.3.3 ASP.NET MVC

3.4 Development Tools

4 Deploying ASP.NET 5 on Windows

4.1 Getting Started

4.2 Deploying ASP.NET 5

4.2.1 Installing .NET
4.2.2 Installing DNVM
4.2.3 Installing git
4.3 Testing
4.3.1 ASP.NET Console
4.3.2 ASP.NET 5
4.3.3 ASP.NET MVC
4.4 Development Tools
5. ASP.NET 5 Development
5.1 Getting Started
5.2 Building ASP.NET 5 Console Application
5.2.1 Creating A Project
5.2.2 Writing Program
5.2.3 Testing
5.3 Building ASP.NET 5 MVC Application
5.3.1 Creating a Project
5.3.2 Writing Program
5.3.2.1 Creating Model
5.3.2.2 Creating Controller
5.3.2.3 Creating View
5.3.3 Testing
6. ASP.NET 5 RESTful and AngularJS
6.1 Getting Started
6.2 Creating Project
6.3 Writing Program
6.3.1 Creating Model
6.3.2 Dummy Database
6.3.3 Creating Controller
6.3.4 Creating View
6.3.5 AngularJS Program
6.4 Testing

6.5 What’s The Next?

Source Code

Contact

Preface

This book was written to help anyone want to get started with ASP.NET 5 core. It describes ASP.NET 5 core deploying and the basic elements of ASP.NET 5 core and shows how to work with several ASP.NET 5 core features.

Agus Kurniawan

Berlin & Depok, June 2015

1. Preparing Development Environment

1.1 ASP.NET 5

ASP.NET 5 is a lean and composable framework for building web and cloud applications. ASP.NET 5 is fully open source and available on GitHub, http://github.com/aspnet/home . ASP.NET 5 is currently in preview when this book is written.

This book will focus on how to develop ASP.NET 5 on OS X, Linux and Windows platforms. We use ASP.NET 5 core beta4.

1.2 System Requirements

For testing, I use Debian/Ubuntu Linux, OS X Yosemite and Windows 8.1. You can use any Linux distribution which inheritanced from Debian based.

1.3 Development Tools

In general, we can use any editor to write ASP.NET 5program. I recommend to use Sublime Text, http://www.sublimetext.com/ , to write ASP.NET 5 codes. Microsoft also released Visual Studio Code and you can download it on https://code.visualstudio.com/ .

2 Deploying ASP.NET 5 on OS X

In this chapter we will learn how to deploy ASP.NET 5 on OS X.

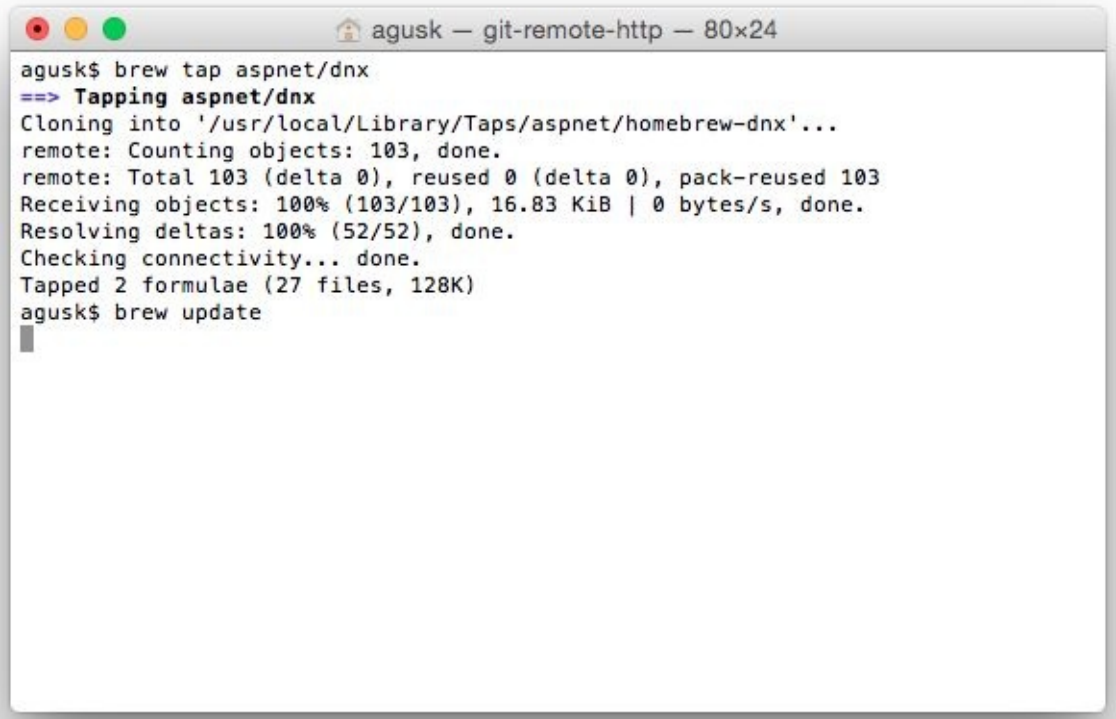
2.1 Getting Started

In this section we start to deploy ASP.NET 5 on OS X. In this book I use ASP.NET 5 preview, beta4. Please follow installation steps on next section.

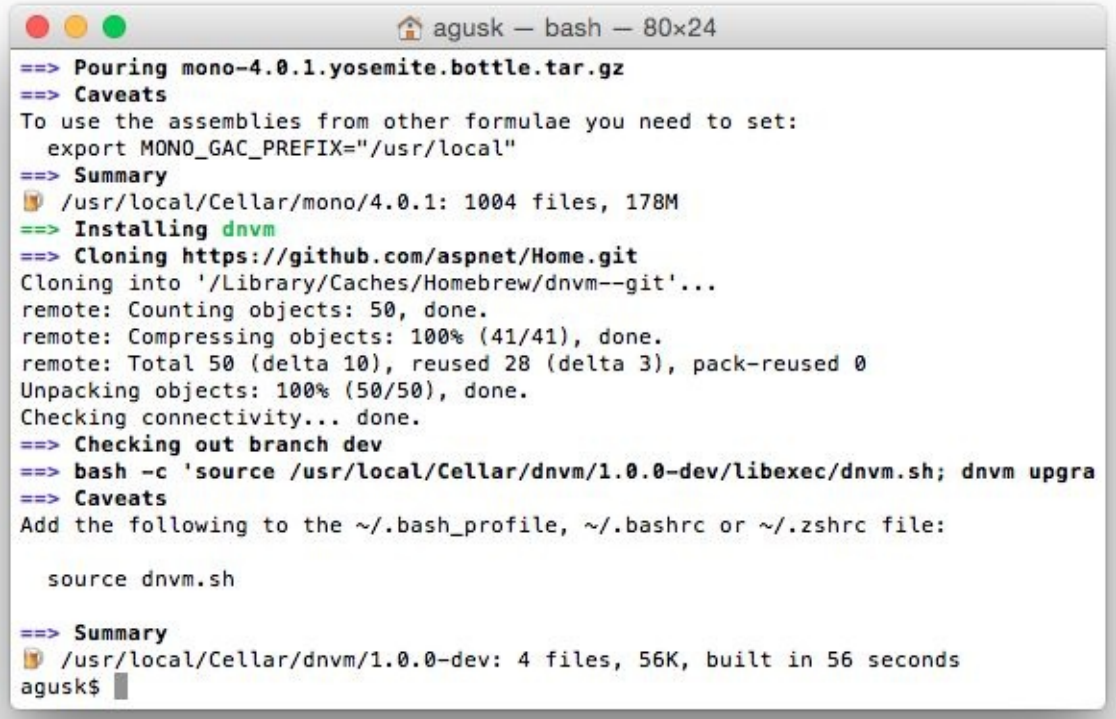
2.2 Deploying ASP.NET 5

To install ASP.NET on OS X, you can use brew, http://brew.sh/ . Type these commands to install ASP.NET 5.

```
$ brew tap aspnet/dnx
$ brew update
$ brew install dnvm
```



If success, you can see a task which you should do. We must add dnvm on our profile.

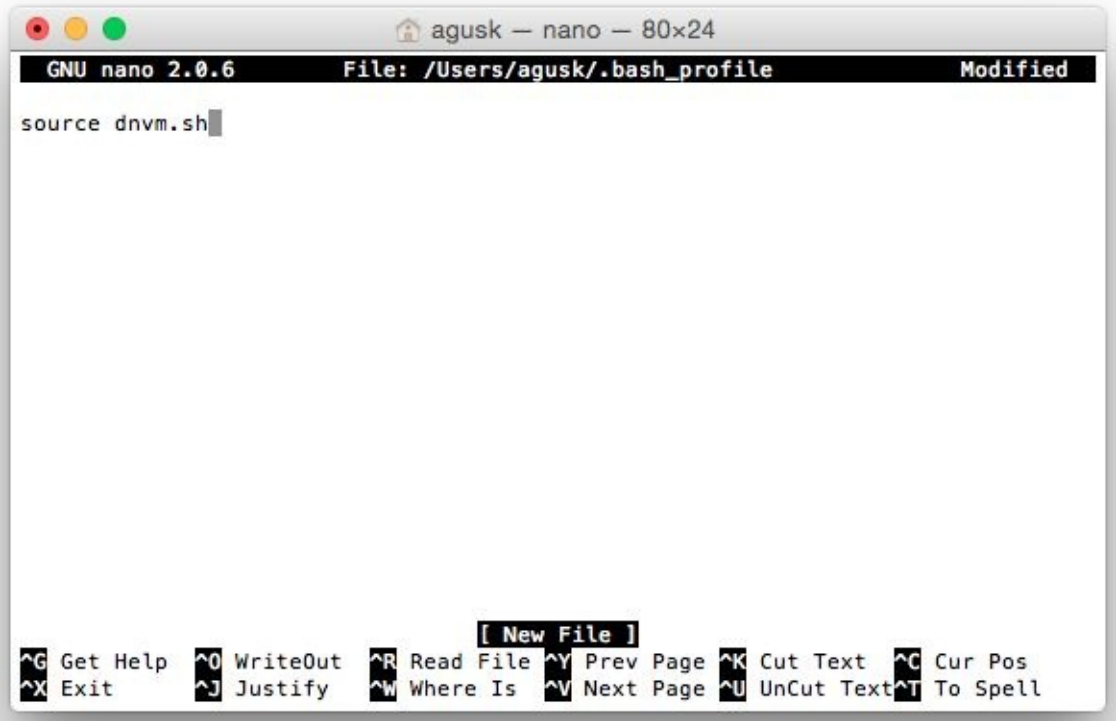


Type this command.

```
$ nano ~/.bash_profile
```

Then, you get nano editor. You can use another text editor. Add this script on ~/.bash_profile.

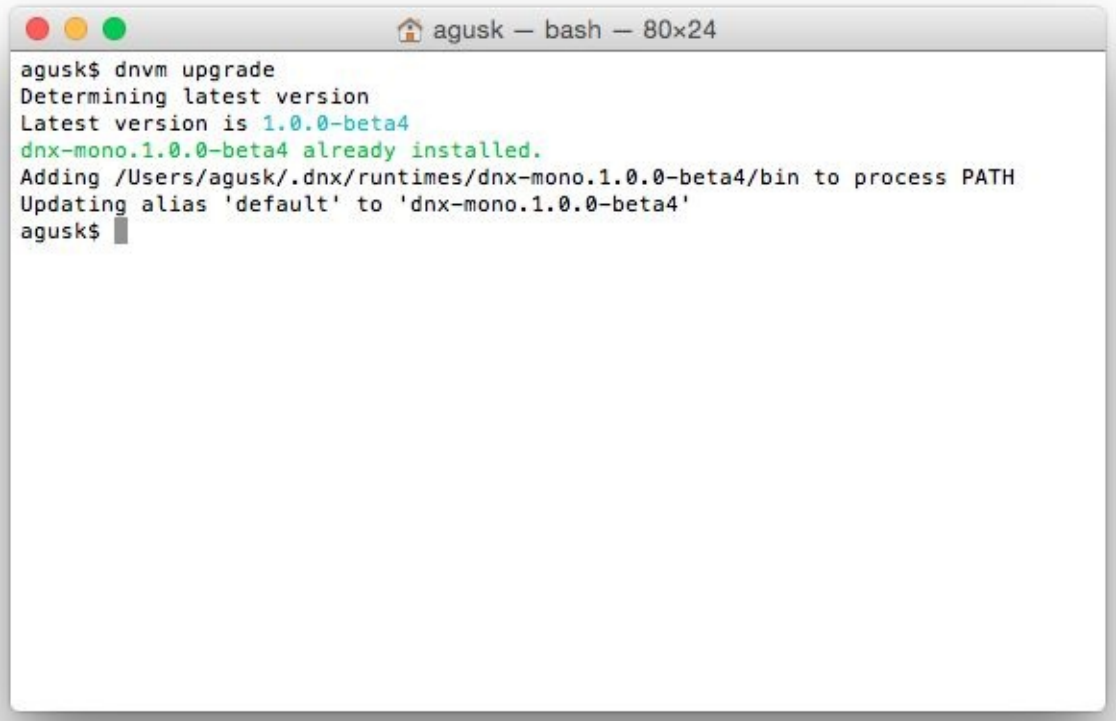
```
source dnvm.sh
```



If done, save this file. After that, close your Terminal and then open again so Terminal will load profile script.

The next step is to upgrade. Type this command.

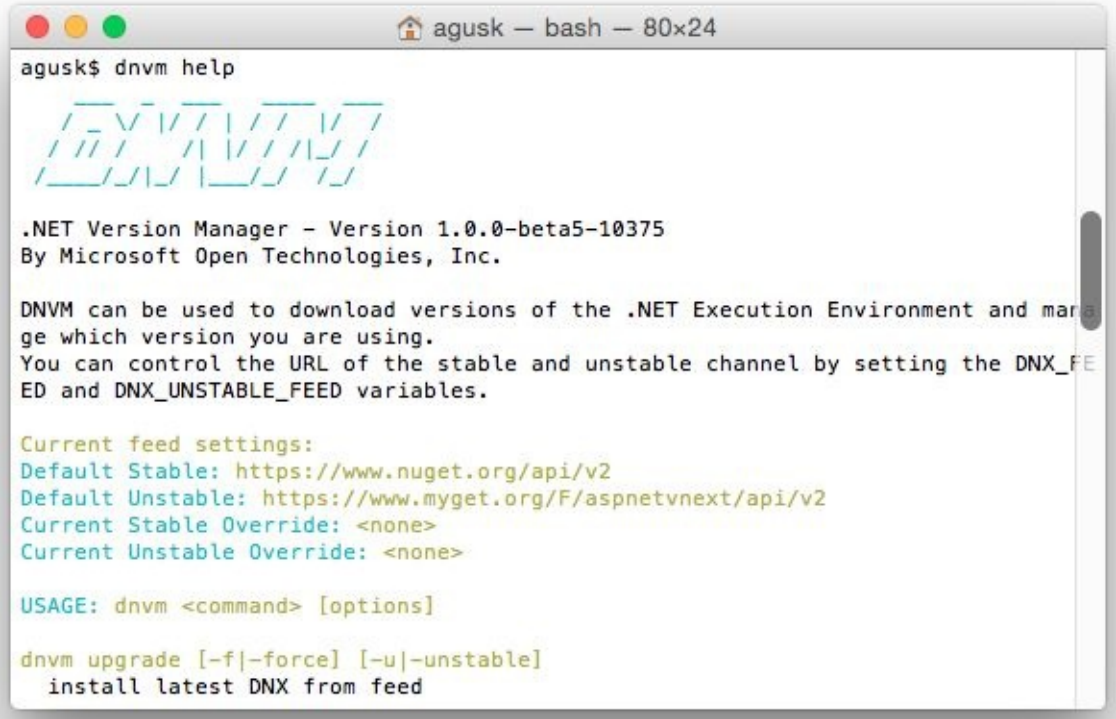
```
$ dnvm upgrade
```



Now you already installed dnvm. It will be used to manage ASP.NET 5 runtime on OS X. You can verify dnvm by typing this command.

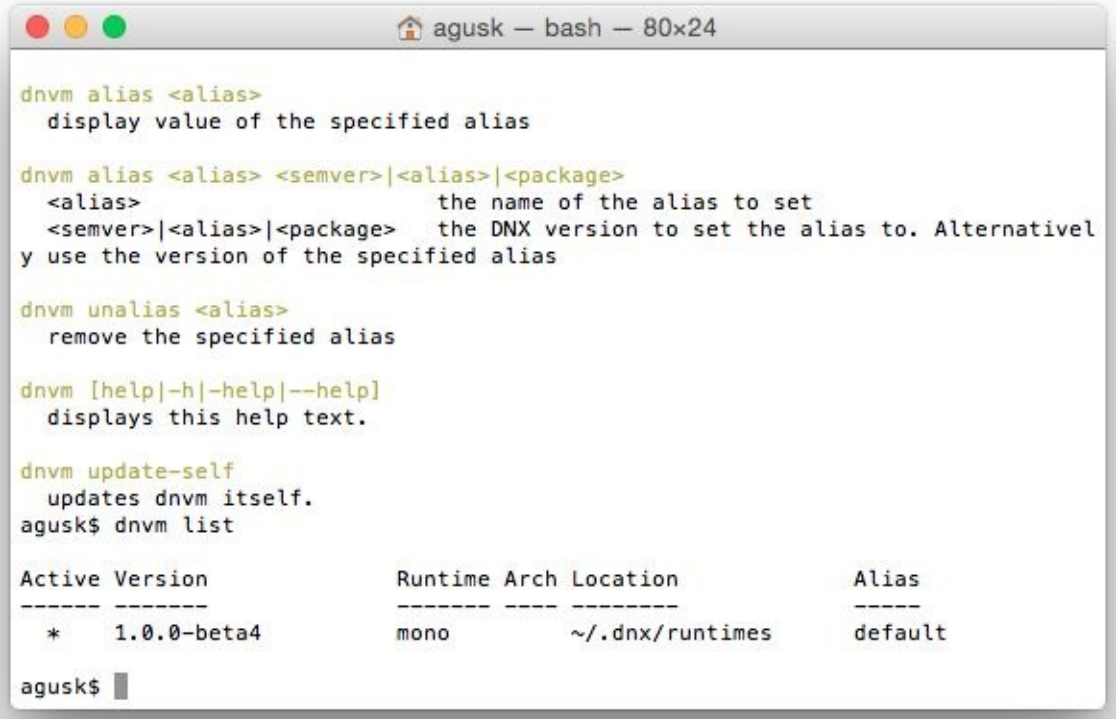
```
$ dnvm help
```

So you see a list of dnvm parameter.



You can also see a list of your DNX runtime by typing this command.

```
$ dnvm list
```


agusk — bash — 80×24
dnvm alias <alias>
display value of the specified alias
dnvm alias <alias> <semver>|<alias>|<package>
<alias> the name of the alias to set
<semver>|<alias>|<package> the DNX version to set the alias to. Alternatively use the version of the specified alias
dnvm unalias <alias>
remove the specified alias
dnvm [help|-h|-help|--help]
displays this help text.
dnvm update-self
updates dnvm itself.
agusk$ dnvm list
Active Version Runtime Arch Location Alias
------ ------- ------- ---- -------- -----
* 1.0.0-beta4 mono ~/.dnx/runtimes default
agusk$

2.3 Testing

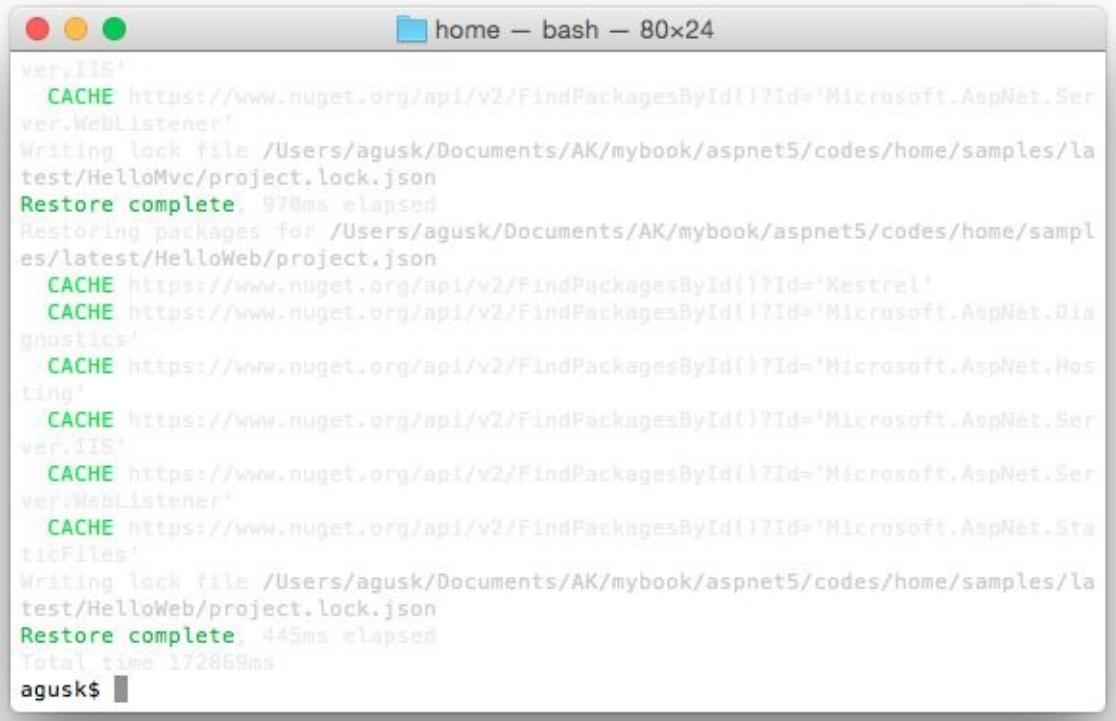
To test our ASP.NET 5, we use code samples from https://github.com/aspnet/Home . We clone this code samples and then restore all required libraries using dnu command.

Type these commands.

```
$ git clone https://github.com/aspnet/Home
$ cd Home
$ dnu restore
```

If you don't git on your OS X, you can install it via brew.

```
$ brew install git
```



Now you are ready to run code samples.

2.3.1 ASP.NET Console

To run ASP.NET Console, we can use dnx. Firstly, navigate to ConsoleApp project.

Type these commands.

```
$ cd samples/latest/ConsoleApp/
$ dnx . run .
```

If success, you can see a message “Hello World” on Terminal.



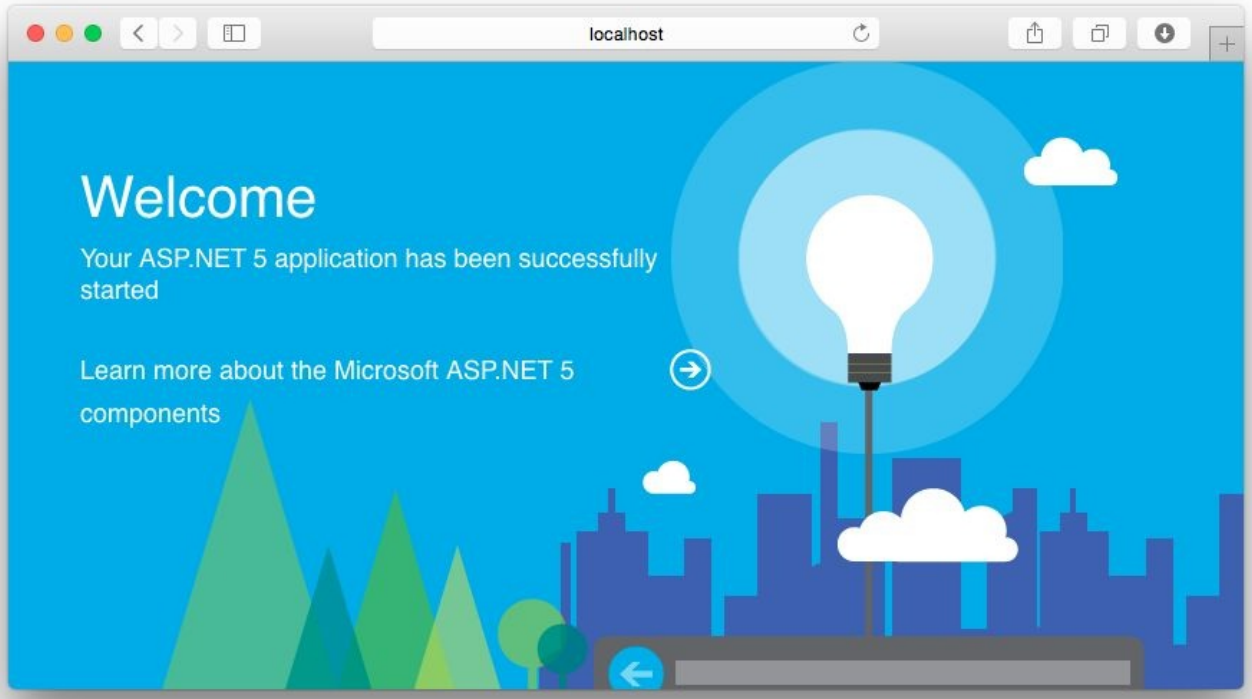
2.3.2 ASP.NET 5

To run ASP.NET 5 app, we can use dnx with kestrel. Firstly, navigate to HelloWeb project.

Type these commands.

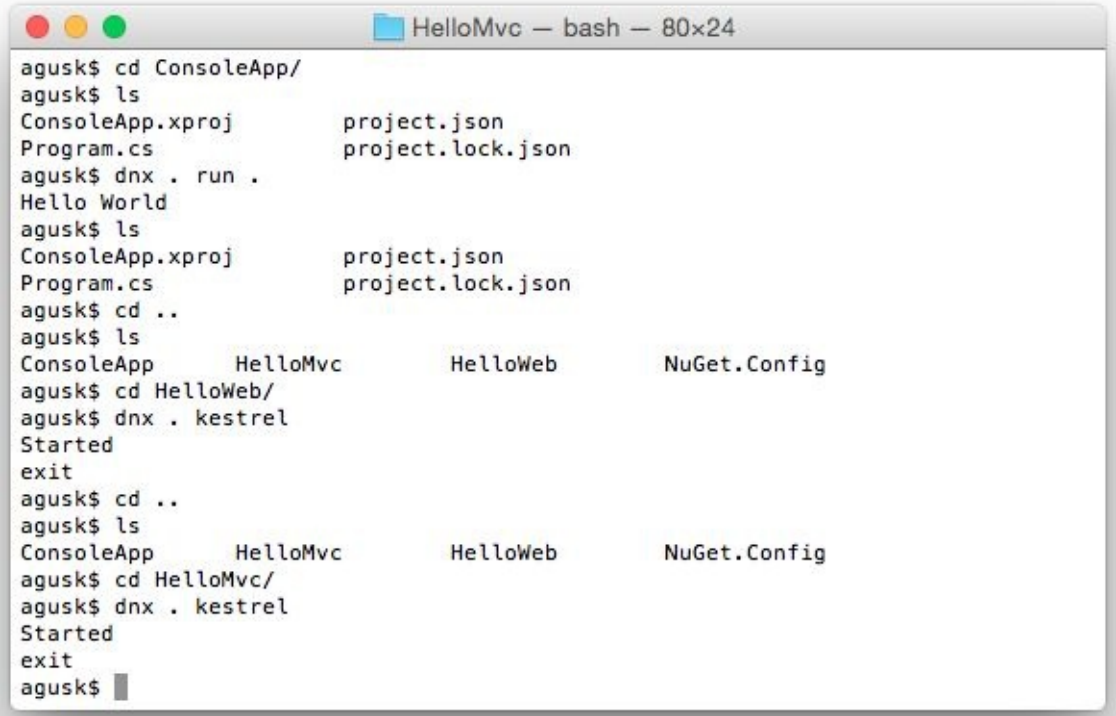
```
$ cd samples/latest/HelloWeb/
$ dnx . kestrel
```

Now open a browser and then navigate to http://localhost:5004/ . You should see ASP.NET 5 app.



To stop kestrel, you can type exit.

```
exit
```



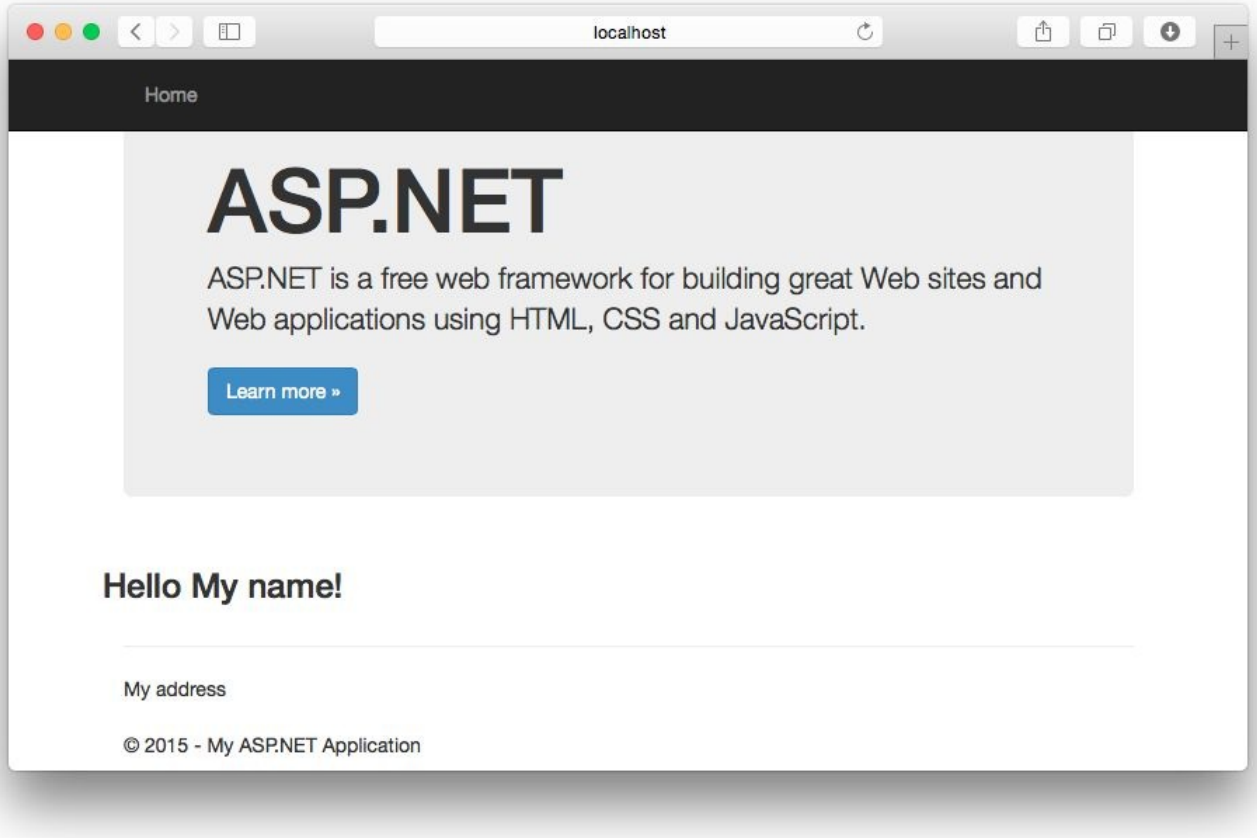
2.3.3 ASP.NET MVC

To run ASP.NET MVC, we can use dnx. Firstly, navigate to HelloMvc project.

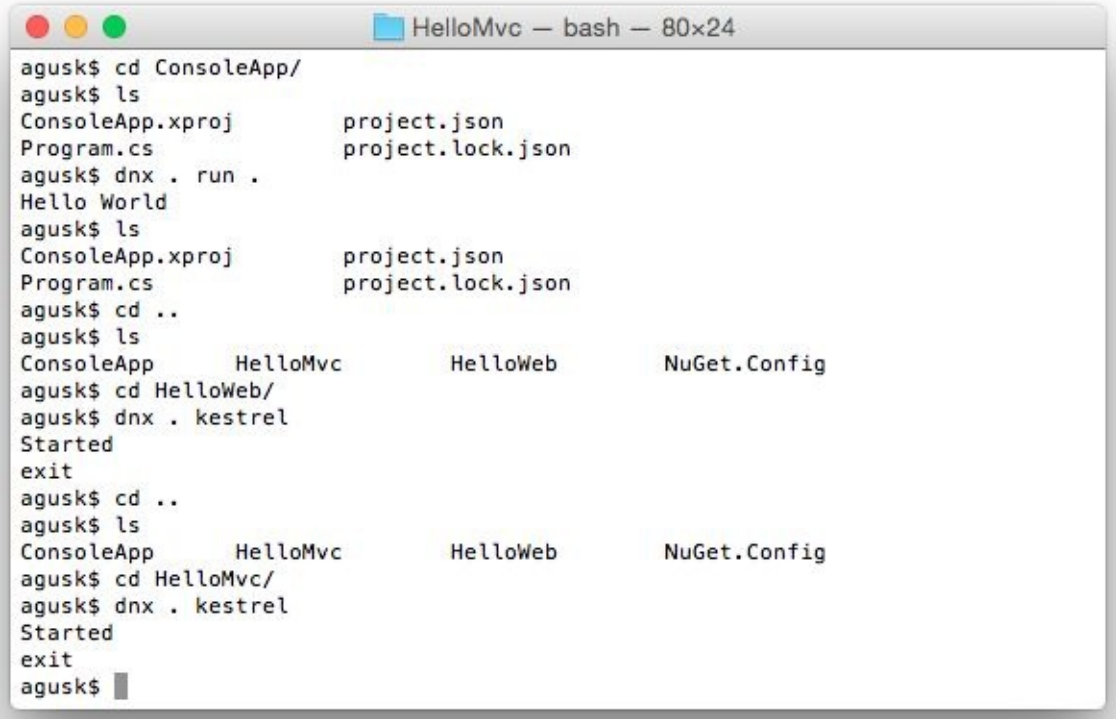
Type these commands.

```
$ cd samples/latest/HelloMvc/
$ dnx . kestrel
```

Now open a browser and then navigate to http://localhost:5004/ . You should see ASP.NET MVC app.

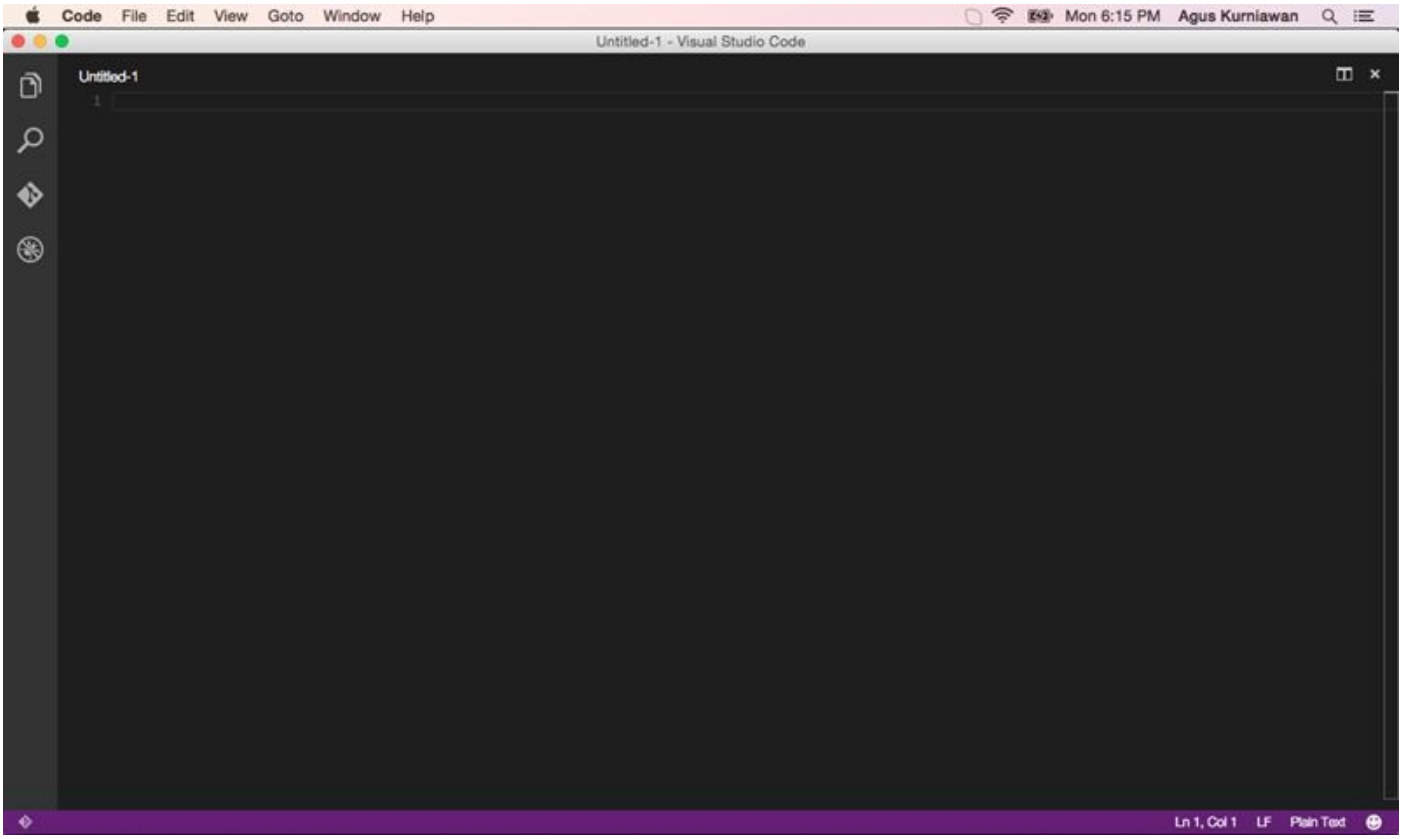


To stop kestrel, you can type exit.


HelloMvc — bash — 80×24
agusk$ cd ConsoleApp/
agusk$ ls
ConsoleApp.xproj project.json
Program.cs project.lock.json
agusk$ dnx . run .
Hello World
agusk$ ls
ConsoleApp.xproj project.json
Program.cs project.lock.json
agusk$ cd ..
agusk$ ls
ConsoleApp HelloMvc HelloWeb NuGet.Config
agusk$ cd HelloWeb/
agusk$ dnx . kestrel
Started
exit
agusk$ cd ..
agusk$ ls
ConsoleApp HelloMvc HelloWeb NuGet.Config
agusk$ cd HelloMvc/
agusk$ dnx . kestrel
Started
exit
agusk$

2.4 Development Tools

The next step is to build program. To write a program, you can use any text editor. Microsoft also provides Visual Studio Code. It runs on Windows, Linux and Mac. You can download it on https://code.visualstudio.com/ .



3 Deploying ASP.NET 5 on Linux

In this chapter we will learn how to deploy ASP.NET 5 on Linux.

3.1 Getting Started

In this section we start to deploy ASP.NET 5 on Linux. In this book I use ASP.NET 5 preview, beta4. For illustration, I use Ubuntu 15.04 as Linux distro. Please follow installation steps on next section.

3.2 Deploying ASP.NET 5

This installation process is based on this documentation, https://github.com/aspnet/Home/blob/dev/GettingStartedDeb.md . In this section, we install ASP.NET 5 on Linux manually (without docker). Firstly, you update your distro.

```
$ sudo apt-get update
```

In general, to install ASP.NET 5 on Ubuntu, you can do the following steps:

- Installing Mono
- Installing libuv
- Installing DNVM
- Installing git (optional)

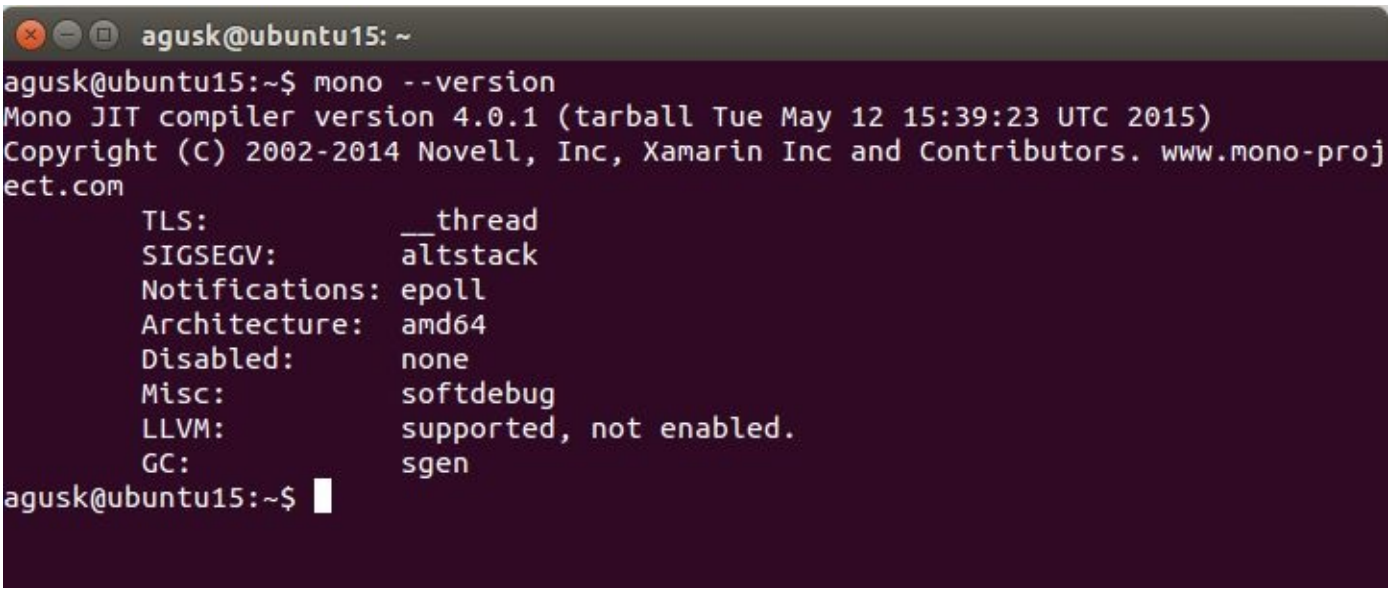
3.2.1 Installing Mono

Mono is .NET runtime for Linux. You need the latest version of Mono runtime. To install it, you can type these commands.

```
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 3FA7E032
$ echo "deb http://download.mono-project.com/repo/debian wheezy main" |
$ sudo apt-get update
$ sudo apt-get install mono-complete
```

If finished, you can verify it by checking its version.

```
$ mono --version
```

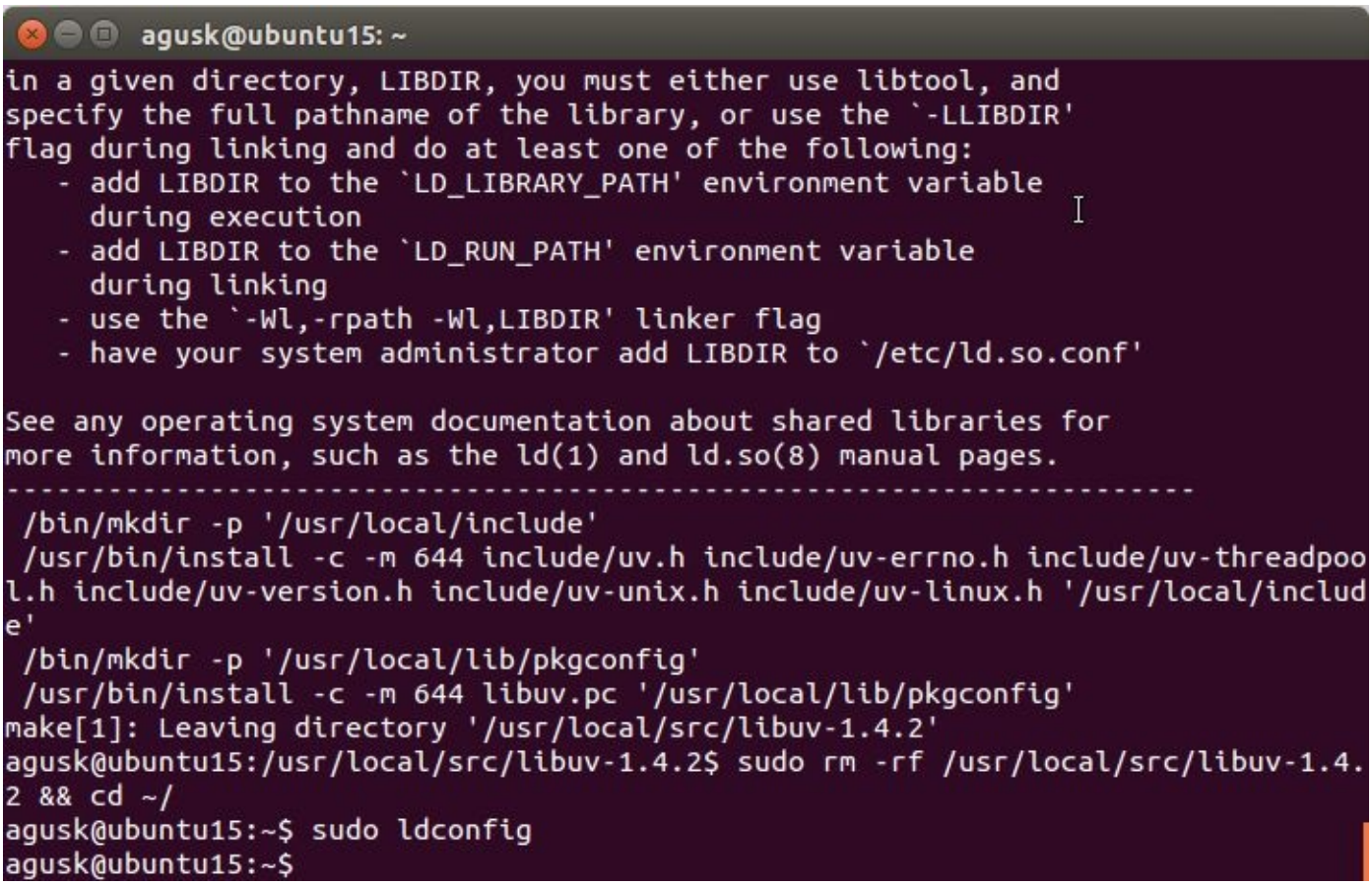


3.2.2 Installing libuv

Libuv is a multi-platform asynchronous IO library that is used by the KestrelHttpServer that we will use to host our web applications.

Type these commands to install libuv library.

```
$ sudo apt-get install automake libtool curl
$ curl -sSL https://github.com/libuv/libuv/archive/v1.4.2.tar.gz | sudo
$ cd /usr/local/src/libuv-1.4.2
$ sudo sh autogen.sh
$ sudo ./configure
$ sudo make
$ sudo make install
$ sudo rm -rf /usr/local/src/libuv-1.4.2 && cd ~/
$ sudo ldconfig
```

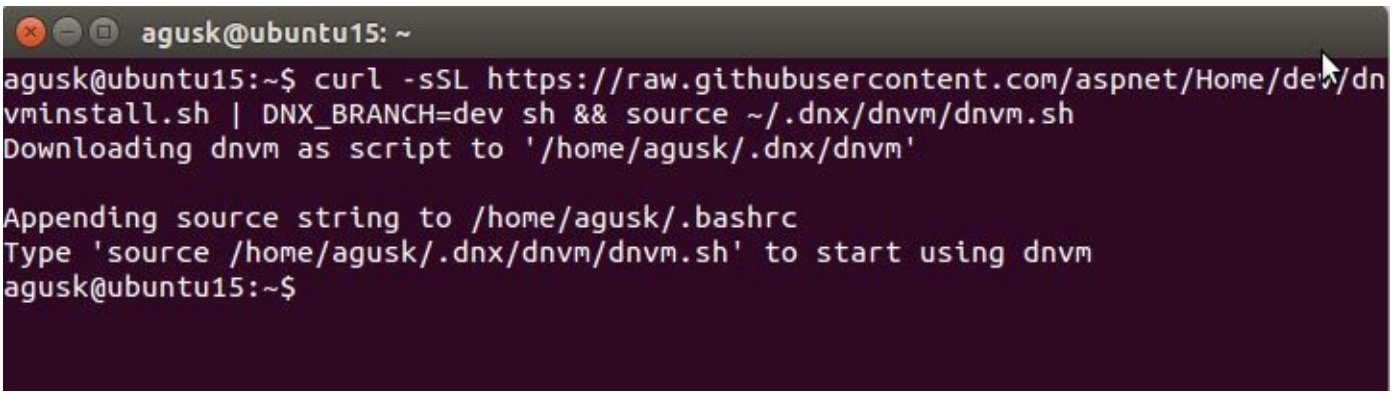


3.2.3 Installing DNVM

The last step is to install DNVM. Type this command.

```
$ curl -sSL https://raw.githubusercontent.com/aspnet/Home/dev/dnvminstal
```

If success, you can see a task which you should do. We must add dnvm on our profile.



Type this command.

```
$ nano /home/<account>/.bashrc
```

Please change <account> by your Linux login account.

Then, you get nano editor. You also edit this file using gedit or other text editor. Add this script on .bashrc file into your profile, for instance, /home/<account>/.bashrc .

Then, type this script.

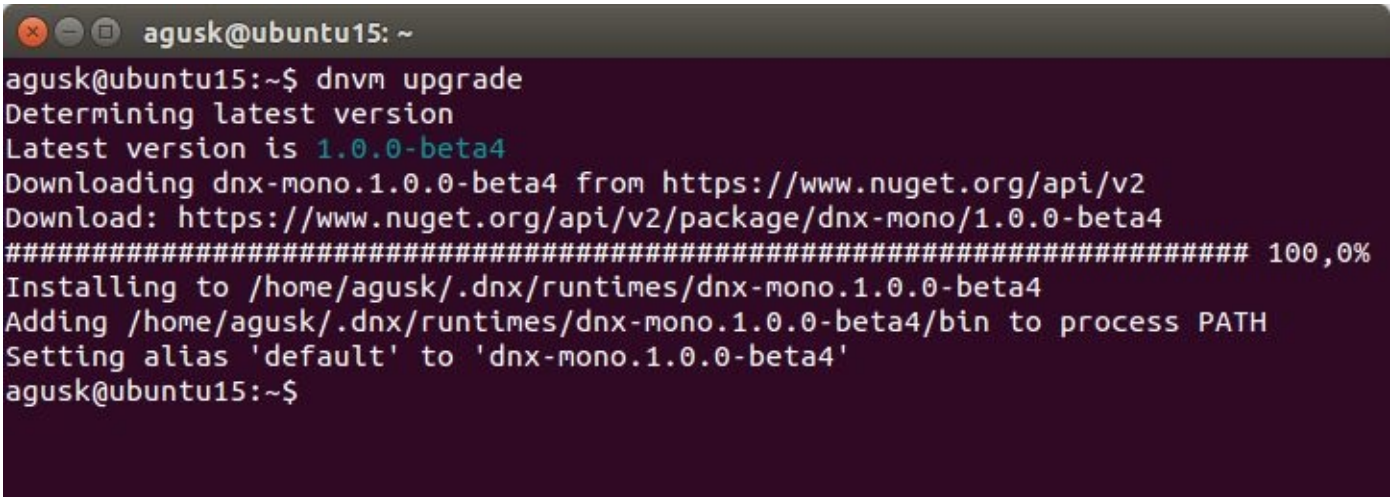
```
source /home/<account>/.dnx/dnvm/dnvm.sh
```

Please change <account> by your Linux login account.

If done, save this file. After that, close your Terminal and then open again so Terminal will load profile script.

The next step is to upgrade. Type this command.

```
$ dnvm upgrade
```



Now you already installed dnvm. It will be used to manage ASP.NET 5 runtime on Linux. You can verify dnvm by typing this command.

```
$ dnvm help
```

So you see a list of dnvm parameter.

```
agusk@ubuntu15: ~
agusk@ubuntu15:~$ dnvm upgrade
Determining latest version
Latest version is 1.0.0-beta4
Downloading dnx-mono.1.0.0-beta4 from https://www.nuget.org/api/v2
Download: https://www.nuget.org/api/v2/package/dnx-mono/1.0.0-beta4
########################################################################## 100,0%
Installing to /home/agusk/.dnx/runtimes/dnx-mono.1.0.0-beta4
Adding /home/agusk/.dnx/runtimes/dnx-mono.1.0.0-beta4/bin to process PATH
Setting alias 'default' to 'dnx-mono.1.0.0-beta4'
agusk@ubuntu15:~$ dnvm help
    ___   _  ___   _____  ___
   / _ \/ |/ / | / / _ |/ _ /
  / // /    /| |/ / /|_/ /
 /____/_/|_/ |___/_/  /_/

.NET Version Manager - Version 1.0.0-beta6-10381
By Microsoft Open Technologies, Inc.

DNVM can be used to download versions of the .NET Execution Environment and mana
ge which version you are using.
You can control the URL of the stable and unstable channel by setting the DNX_FE
ED and DNX_UNSTABLE_FEED variables.

Current feed settings:
```

You can also see a list of your DNX runtime by typing this command.

```
$ dnvm list
```

```
agusk@ubuntu15: ~

dnvm alias <alias>
  display value of the specified alias

dnvm alias <alias> <semver>|<alias>|<package>
  <alias>                       the name of the alias to set
  <semver>|<alias>|<package>    the DNX version to set the alias to. Alternativel
y use the version of the specified alias

dnvm unalias <alias>
  remove the specified alias

dnvm [help|-h|-help|--help]
  displays this help text.

dnvm update-self
  updates dnvm itself.
agusk@ubuntu15:~$ dnvm list

Active Version              Runtime Arch Location            Alias
------ -------              ------- ---- --------            -----
  *    1.0.0-beta4          mono         ~/.dnx/runtimes     default

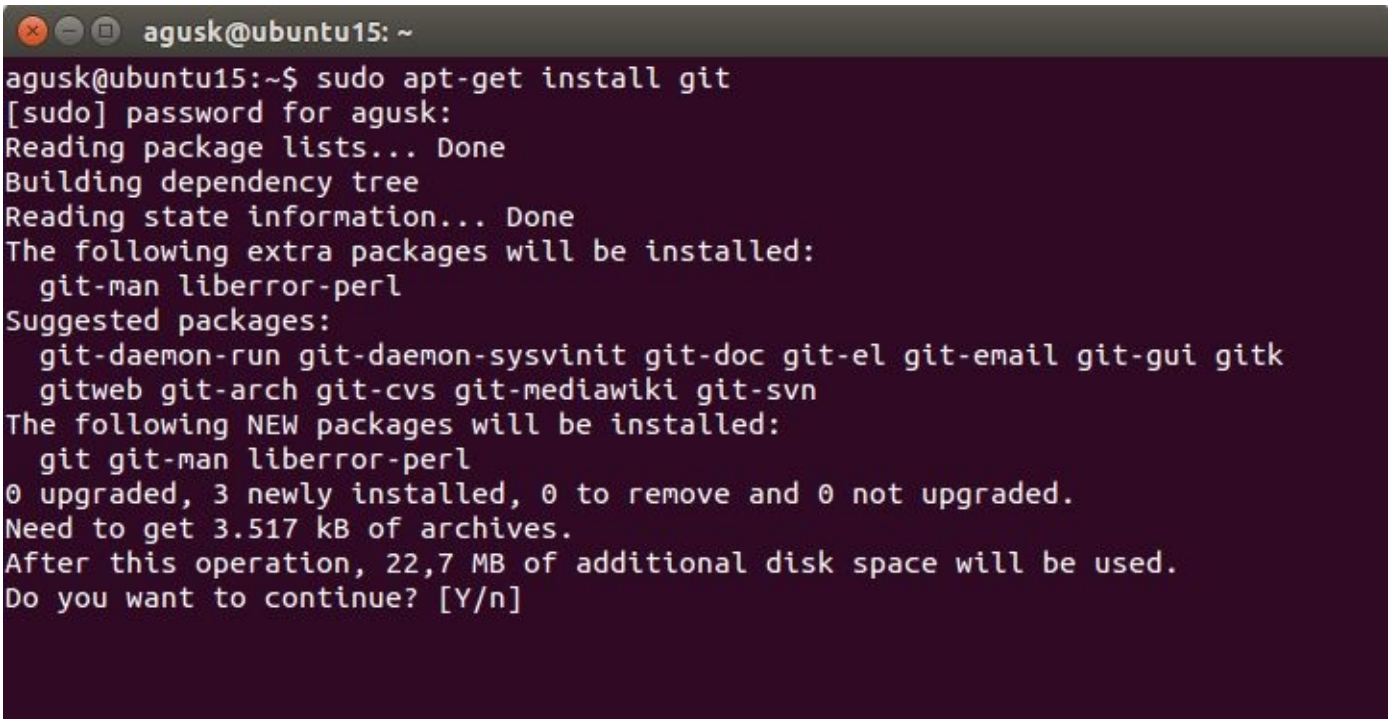
agusk@ubuntu15:~$
```

3.2.4 Installing git

This task is optional but I recommend to install it because we want to run the sample app from https://github.com/aspnet/Home .

Type this command to install git.

```
$ sudo apt-get install git
```

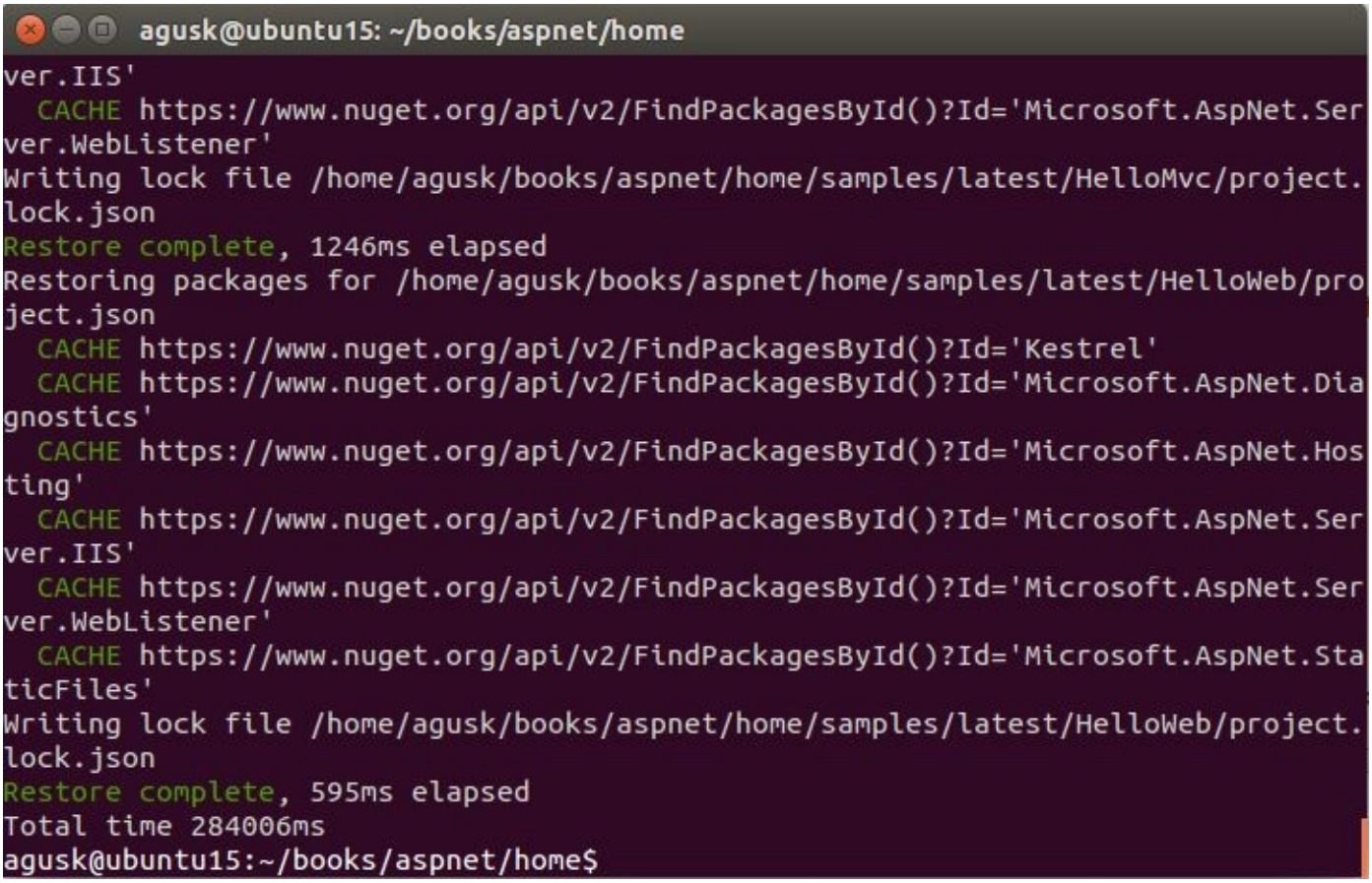


3.3 Testing

To test our ASP.NET 5, we use code samples from https://github.com/aspnet/Home . We clone this code samples and then restore all required libraries using dnu command.

Type these commands.

```
$ git clone https://github.com/aspnet/Home
$ cd Home
$ dnu restore
```



Now you are ready to run code samples.

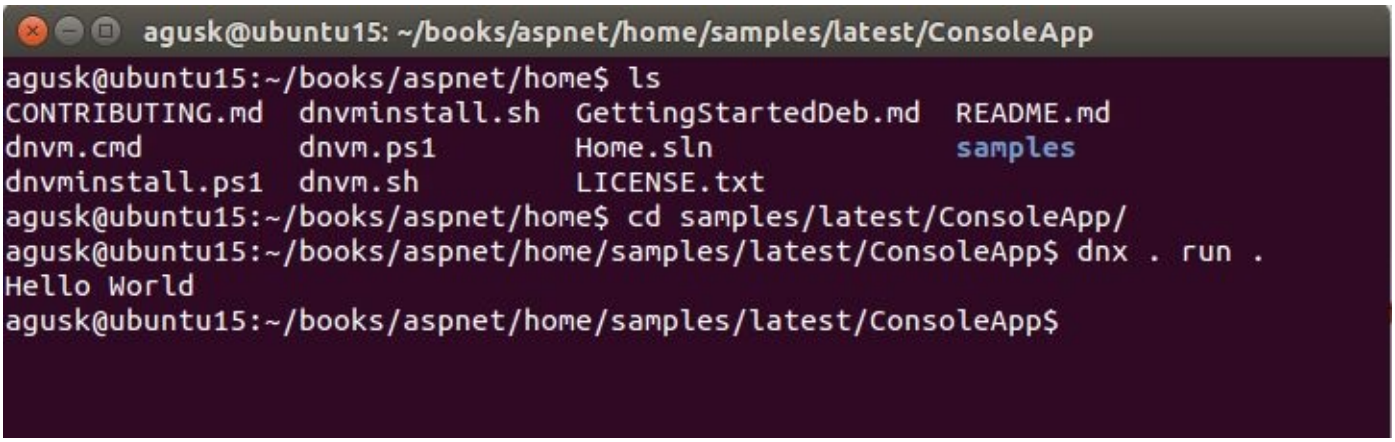
3.3.1 ASP.NET Console

To run ASP.NET Console, we can use dnx. Firstly, navigate to ConsoleApp project.

Type these commands.

```
$ cd samples/latest/ConsoleApp/
$ dnx . run .
```

If success, you can see a message “Hello World” on Terminal.

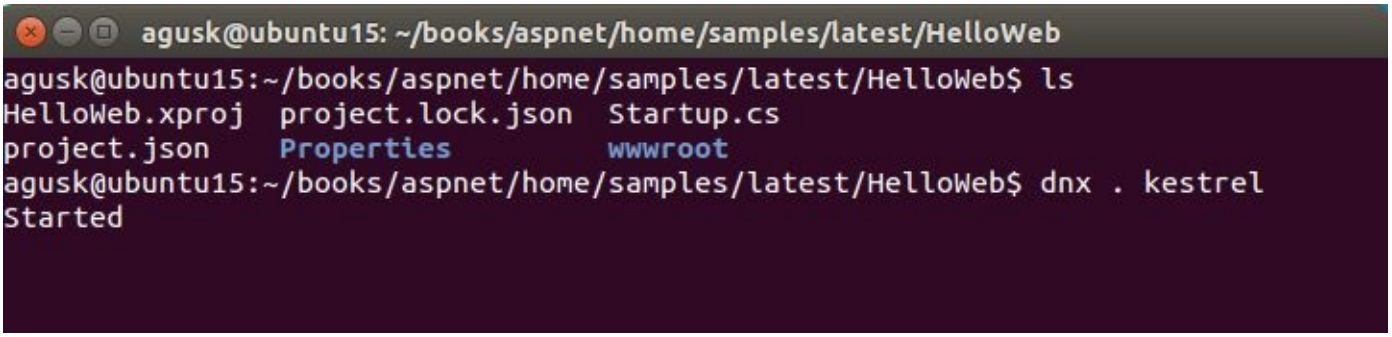


3.3.2 ASP.NET 5

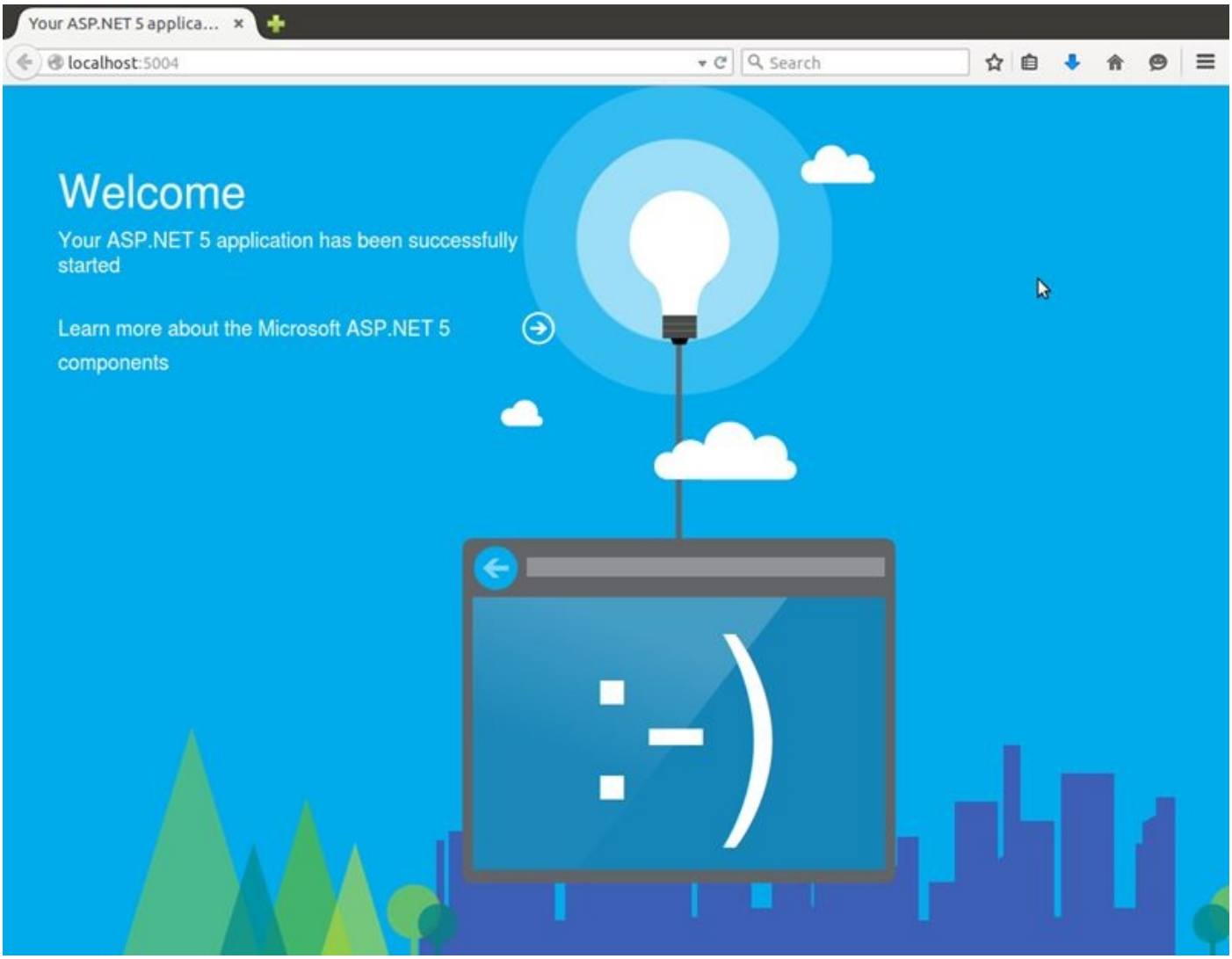
To run ASP.NET 5 app, we can use dnx with kestrel. Firstly, navigate to HelloWeb project.

Type these commands.

```
$ cd samples/latest/HelloWeb/
$ dnx . kestrel
```



Now open a browser and then navigate to http://localhost:5004/ . You should see ASP.NET 5 app.



To stop kestrel, you can type exit.

```
exit
```

3.3.3 ASP.NET MVC

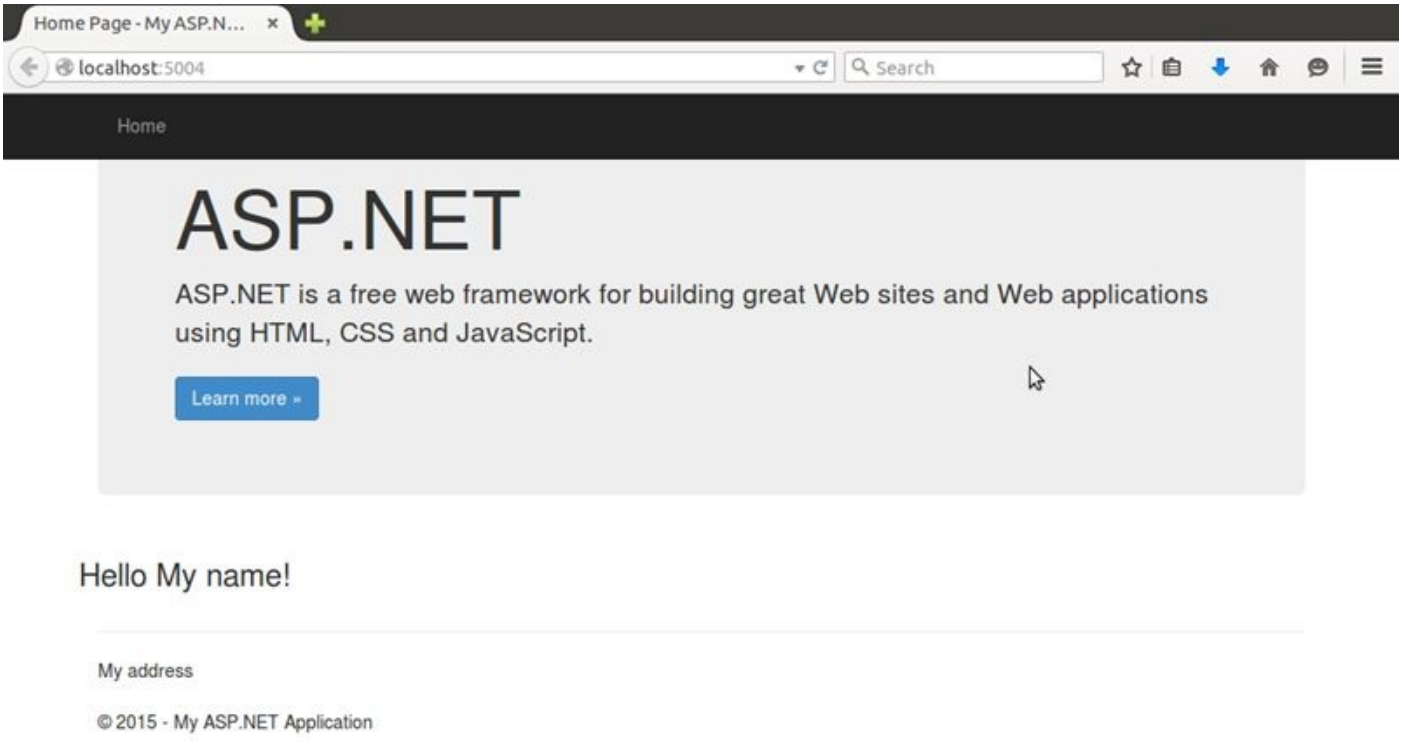
To run ASP.NET MVC, we can use dnx. Firstly, navigate to HelloMvc project.

Type these commands.

```
$ cd samples/latest/HelloMvc/
$ dnx . kestrel
```

```
agusk@ubuntu15: ~/books/aspnet/home/samples/latest/HelloMvc
agusk@ubuntu15:~/books/aspnet/home/samples/latest/HelloMvc$ ls
Controllers     Models        project.lock.json  Startup.cs  wwwroot
HelloMvc.xproj  project.json  Properties         Views
agusk@ubuntu15:~/books/aspnet/home/samples/latest/HelloMvc$ dnx . kestrel
Started
```

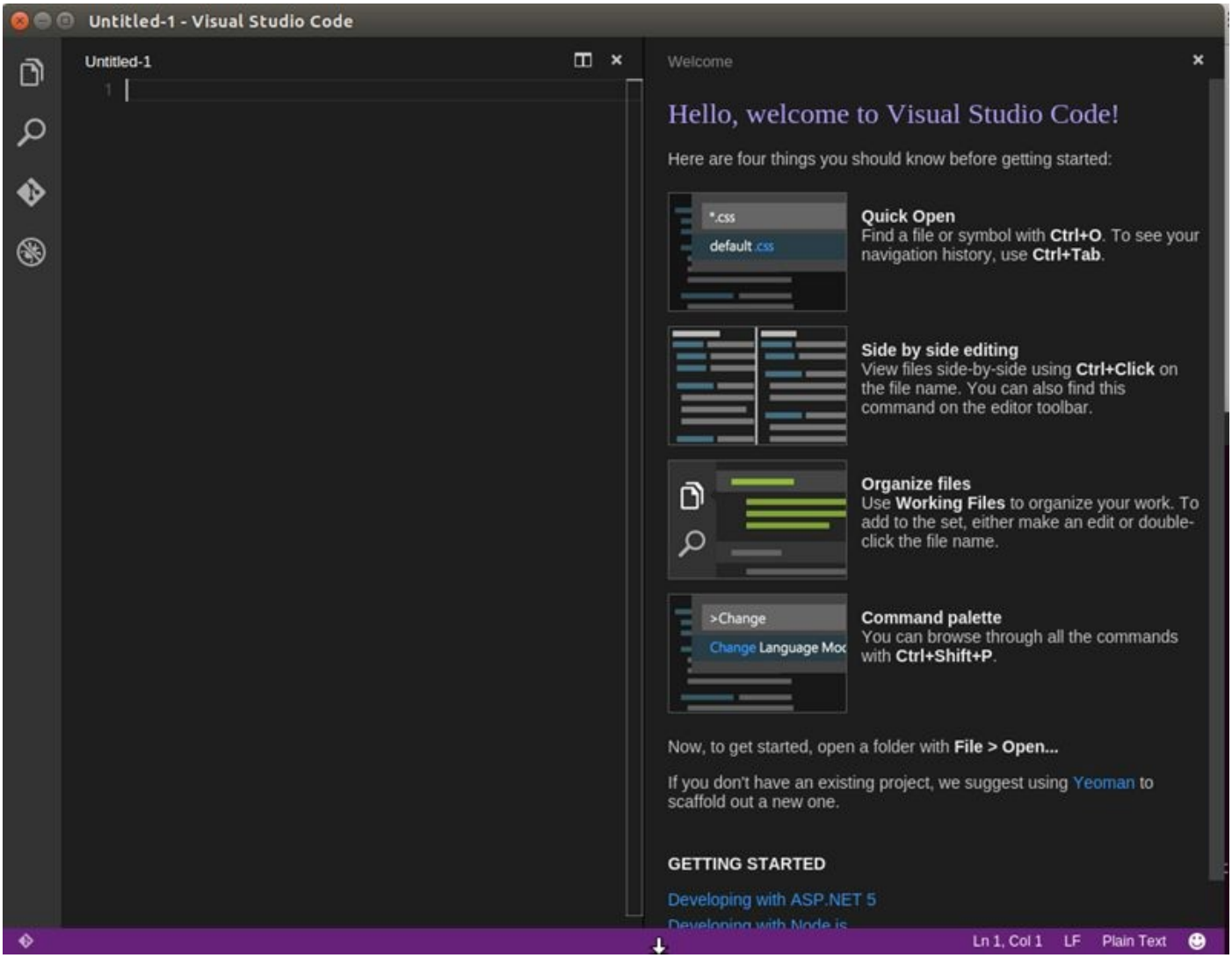
Now open a browser and then navigate to http://localhost:5004/ . You should see ASP.NET MVC app.



To stop kestrel, you can type exit.

3.4 Development Tools

The next step is to build program. To write a program, you can use any text editor. Microsoft also provides Visual Studio Code. It runs on Windows, Linux and Mac. You can download it on https://code.visualstudio.com/ .



4 Deploying ASP.NET 5 on Windows

In this chapter we will learn how to deploy ASP.NET 5 on Windows.

4.1 Getting Started

In this section we start to deploy ASP.NET 5 on Windows. In this book I use ASP.NET 5 preview, beta4. For illustration, I use Windows 8.1. Please follow installation steps on next section.

4.2 Deploying ASP.NET 5

This installation process is based on this documentation, https://github.com/aspnet/Home/blob/dev/GettingStartedDeb.md . In this section, we install ASP.NET 5 on Windows. Firstly, you update your Windows.

In general, to install ASP.NET 5 on Ubuntu, you can do the following steps:

- Installing .NET
- Installing DNVM
- Installing git (optional)

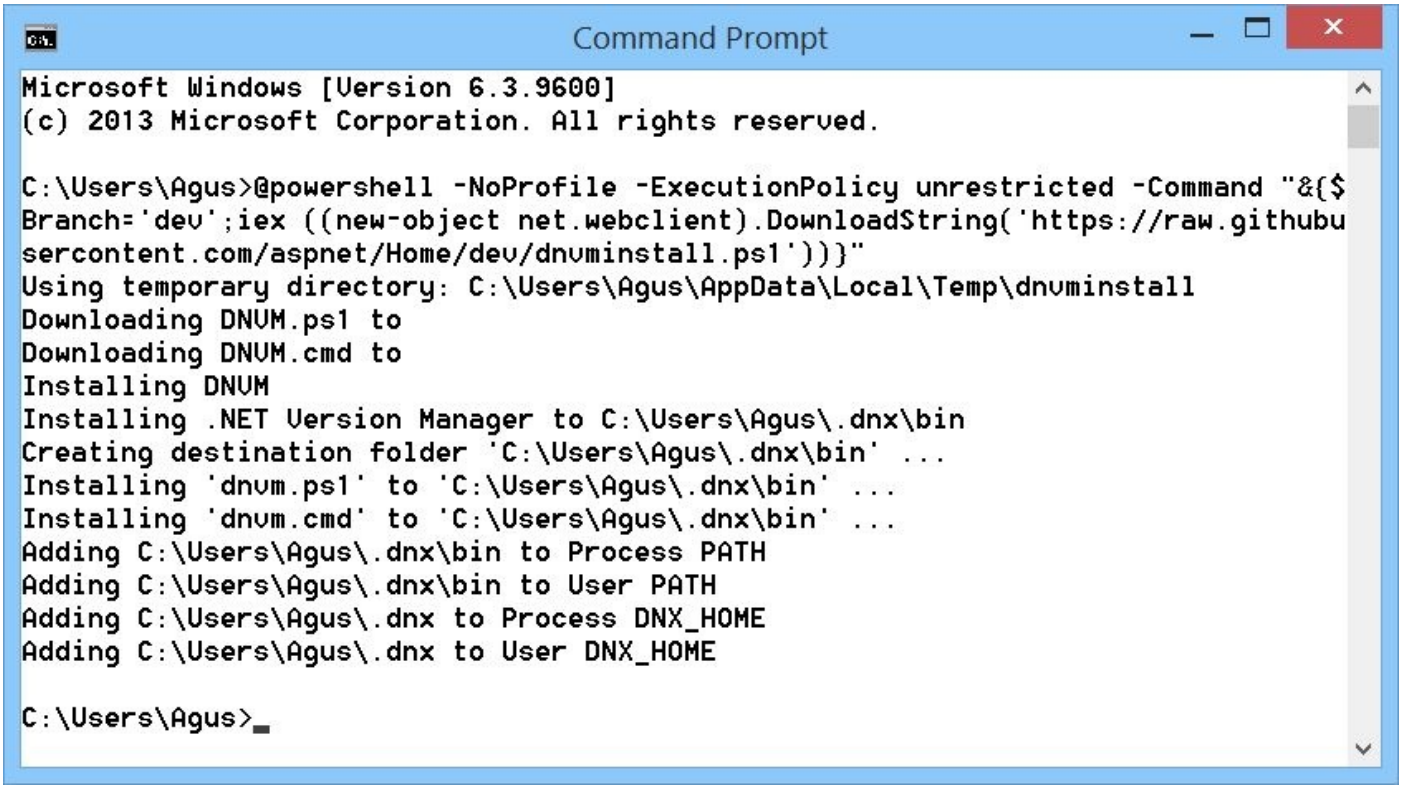
4.2.1 Installing .NET

You need the latest version of .NET Framework runtime. To install it, you can download it on https://www.microsoft.com/net .

4.2.2 Installing DNVM

You can install DNVM via Command Prompt. Open CMD and type this command.

```
@powershell -NoProfile -ExecutionPolicy unrestricted -Command "&{$Branch
```

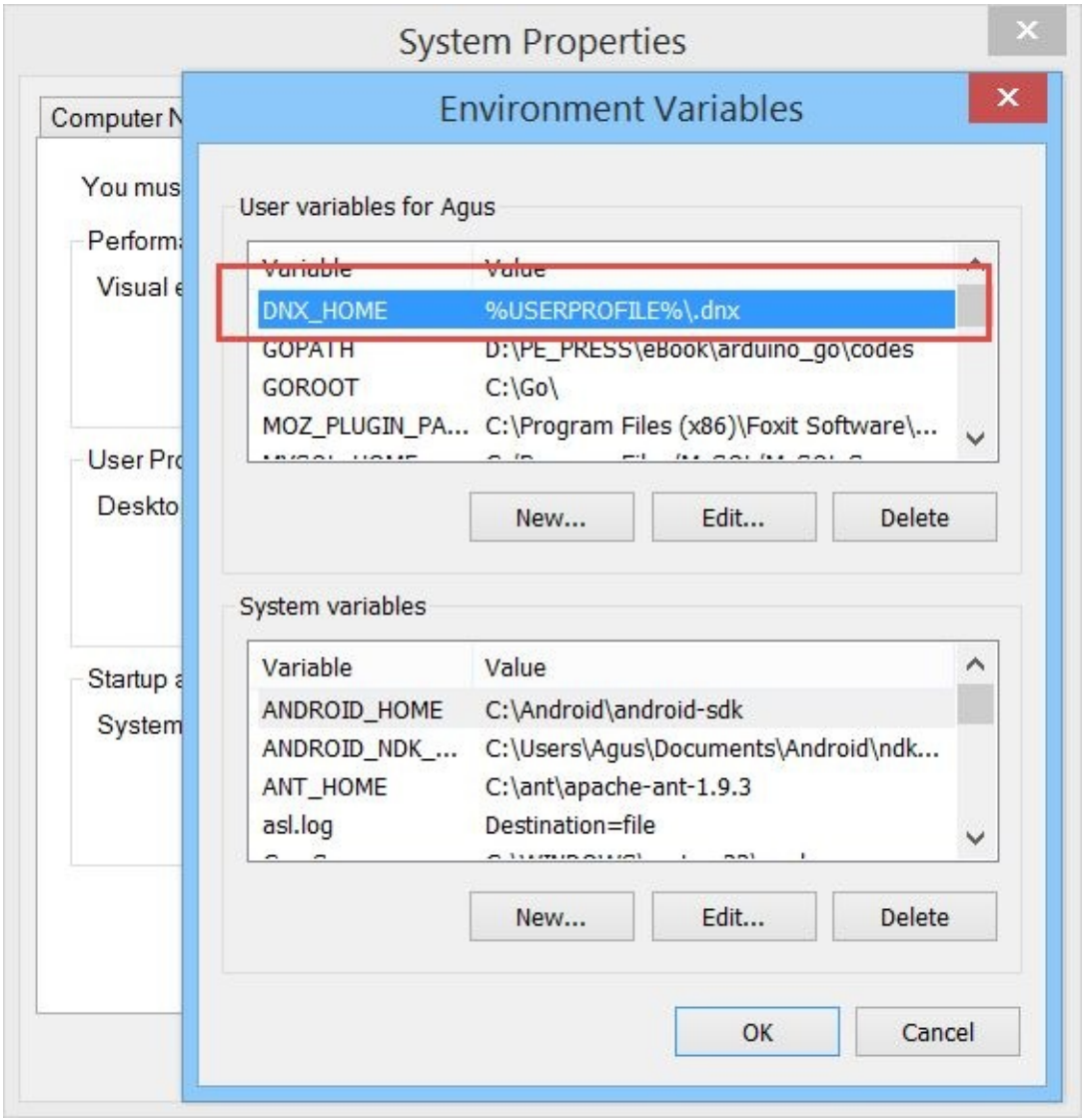


If success, you can verify where dnvm is by typing this command.

```
where dnvm
```

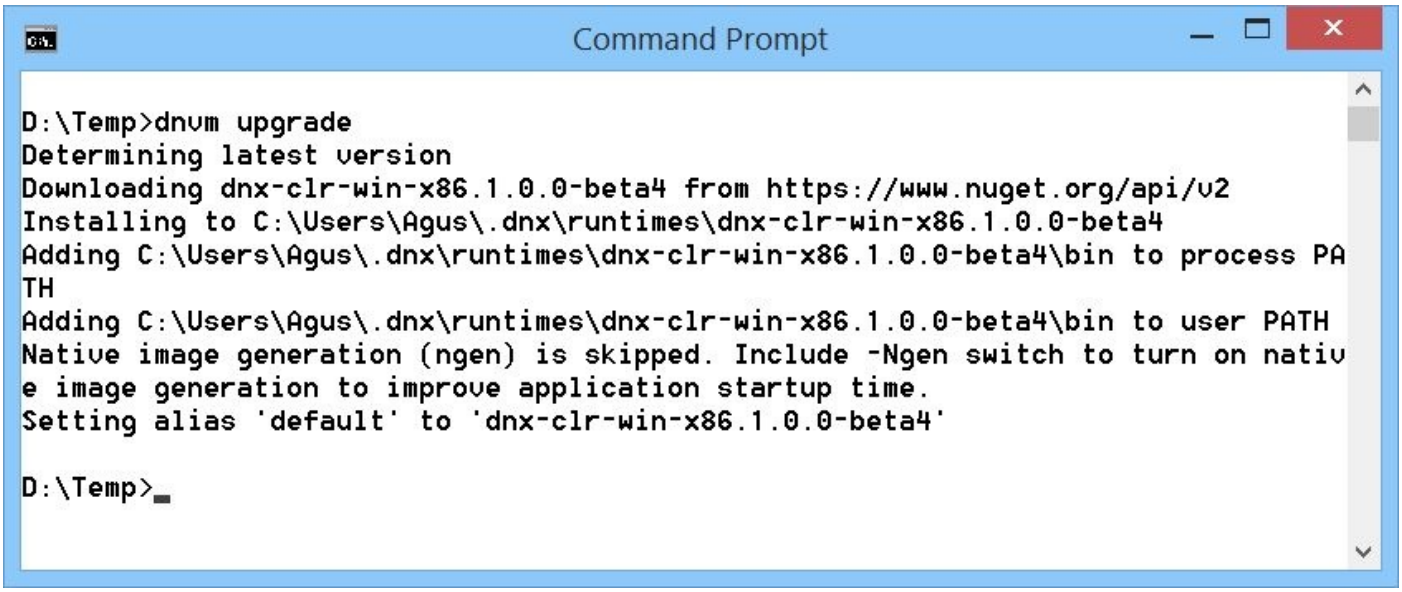


If got error, check your PATH configuration. Make sure .dnx path already added.



The next step is to upgrade DNVM. Type this command.

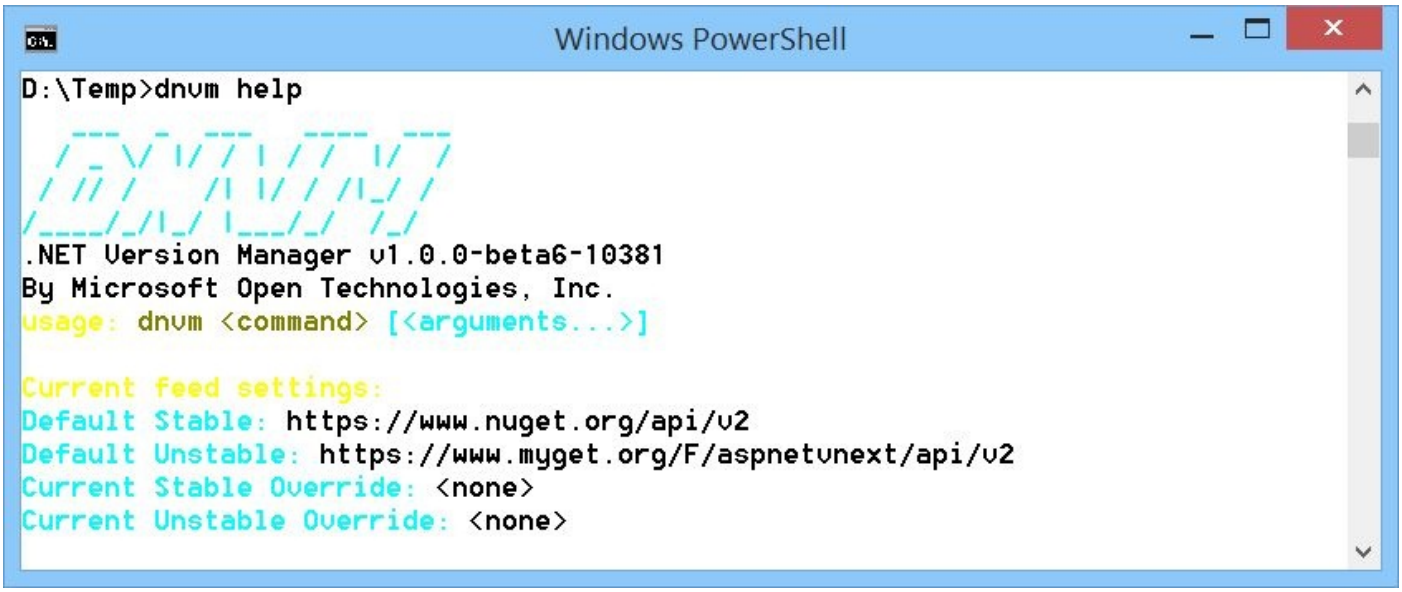
```
$ dnvm upgrade
```



Now you already installed dnvm. It will be used to manage ASP.NET 5 runtime on Windows. You can verify dnvm by typing this command.

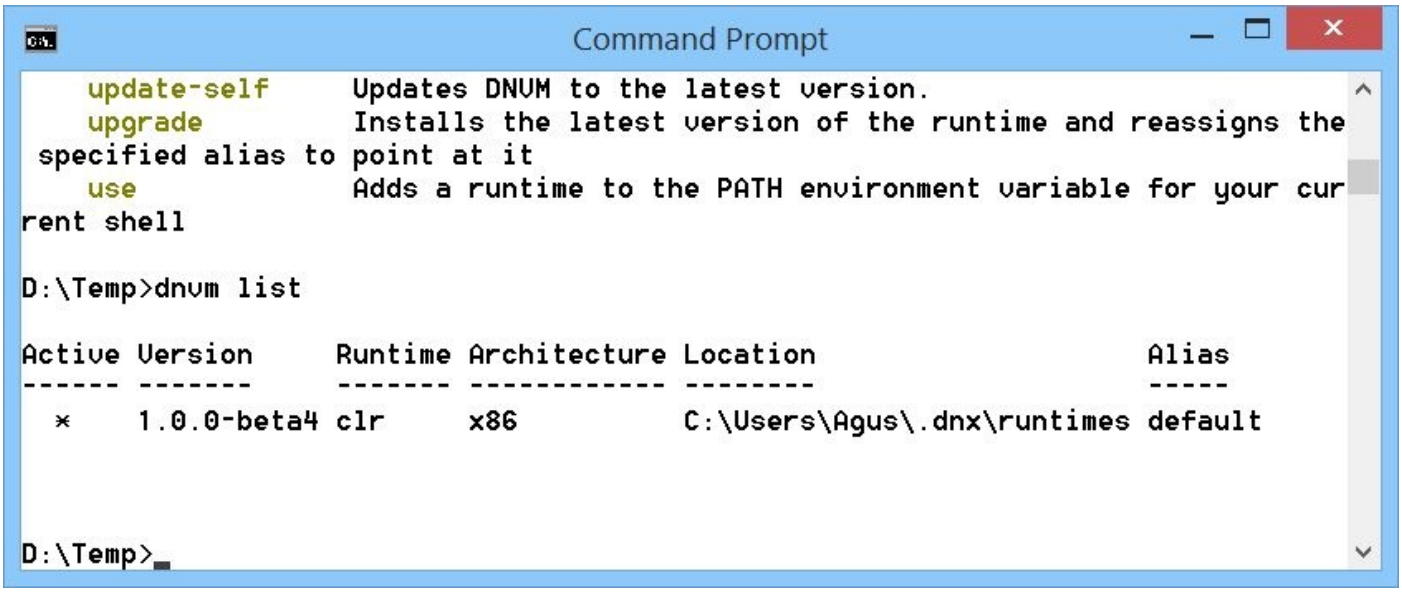
```
$ dnvm help
```

So you see a list of dnvm parameter.



You can also see a list of your DNX runtime by typing this command.

```
$ dnvm list
```



4.2.3 Installing git

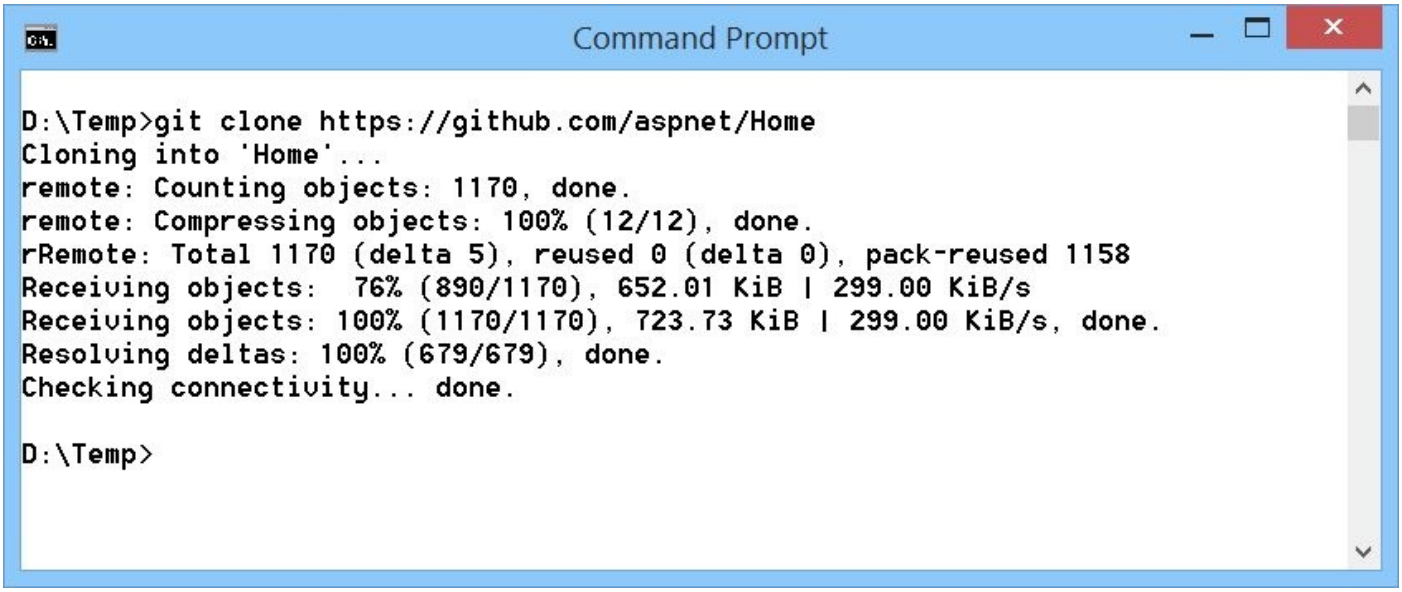
This task is optional but I recommend to install it because we want to run the sample app from https://github.com/aspnet/Home . You can install git by downloading it on https://git-scm.com/downloads .

4.3 Testing

To test our ASP.NET 5, we use code samples from https://github.com/aspnet/Home . We clone this code samples and then restore all required libraries using dnu command.

Type these commands.

```
$ git clone https://github.com/aspnet/Home
$ cd Home
$ dnu restore
```





Now you are ready to run code samples.

4.3.1 ASP.NET Console

To run ASP.NET Console, we can use dnx. Firstly, navigate to ConsoleApp project.

Type these commands.

```
$ cd samples/latest/ConsoleApp/
$ dnx . run .
```

If success, you can see a message “Hello World” on Terminal.

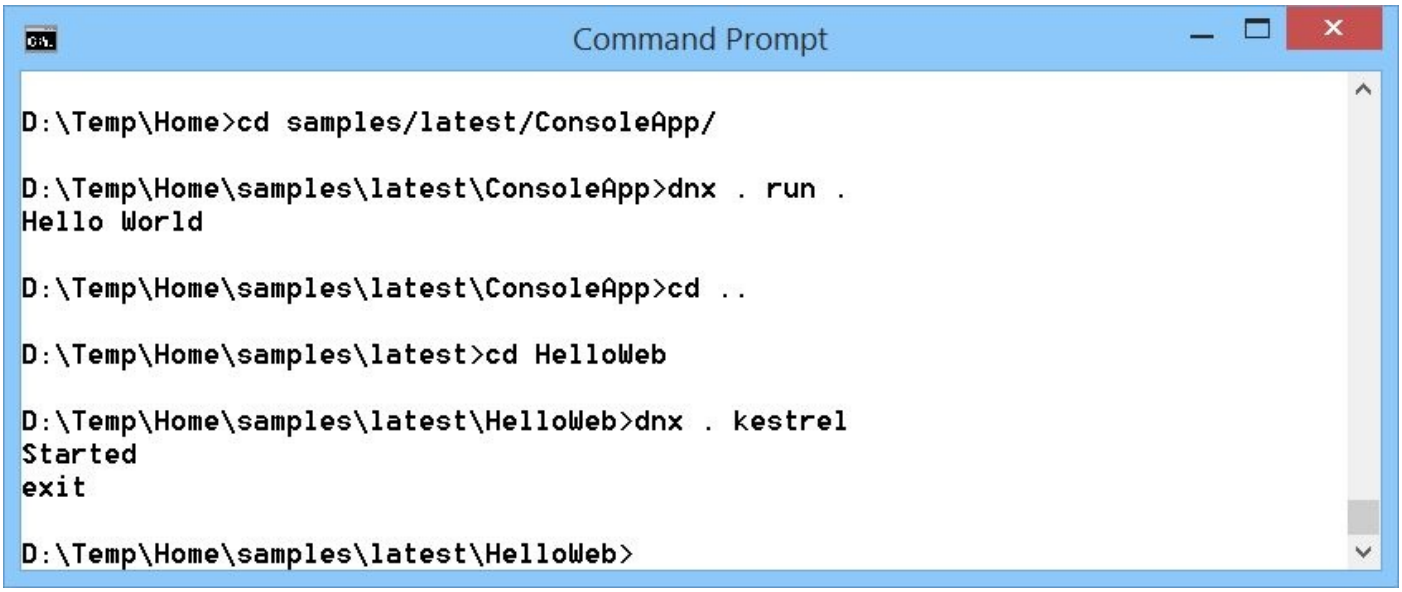


4.3.2 ASP.NET 5

To run ASP.NET 5 app, we can use dnx with kestrel. Firstly, navigate to HelloWeb project.

Type these commands.

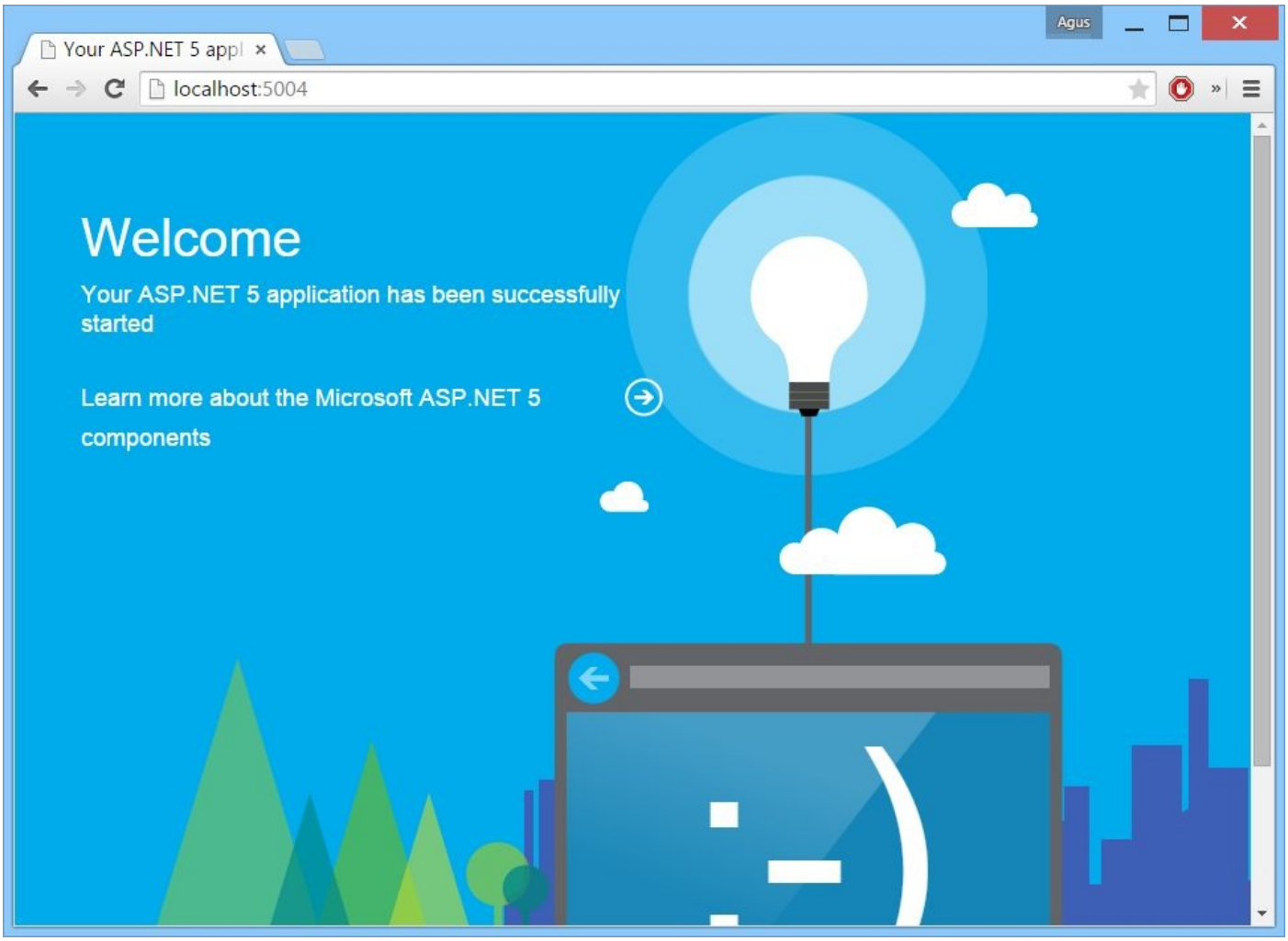
```
$ cd samples/latest/HelloWeb/
$ dnx . kestrel
```



When you run these commands, you may get a security confirmation. Please click **Allow access** button to give a security permission.



Now open a browser and then navigate to http://localhost:5004/ . You should see ASP.NET 5 app.



To stop kestrel, you can type exit.

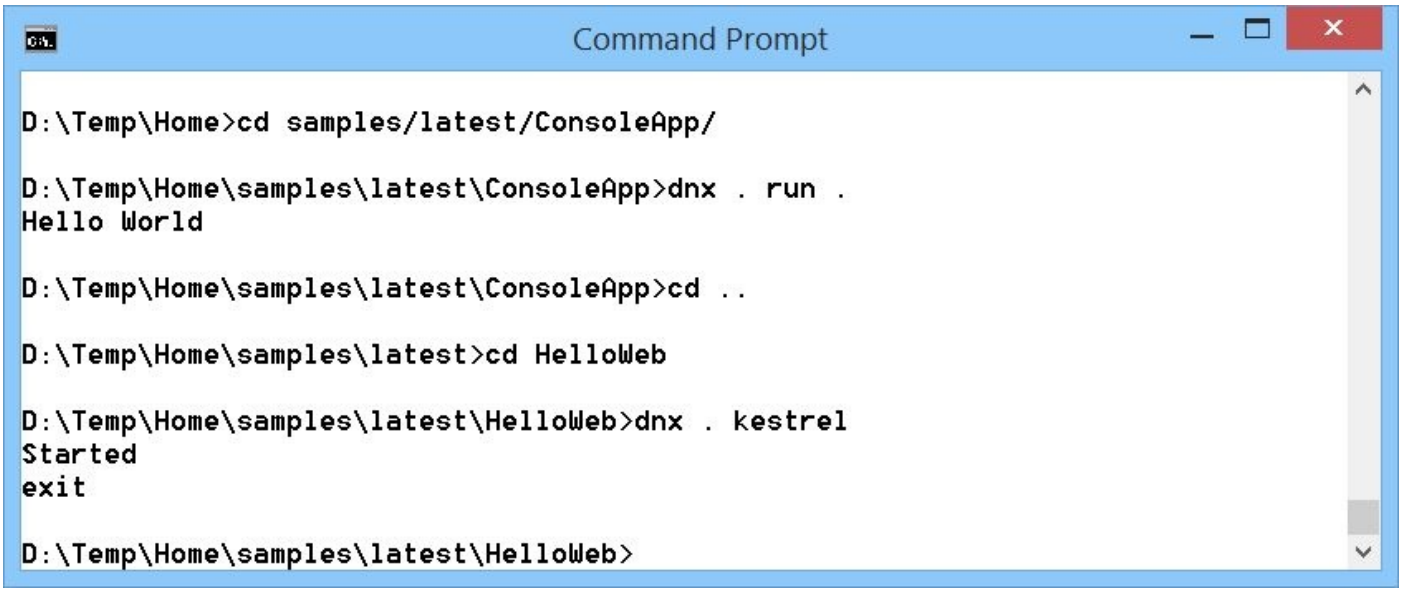
```
exit
```

4.3.3 ASP.NET MVC

To run ASP.NET MVC, we can use dnx. Firstly, navigate to HelloMvc project.

Type these commands.

```
$ cd samples/latest/HelloMvc/
$ dnx . kestrel
```



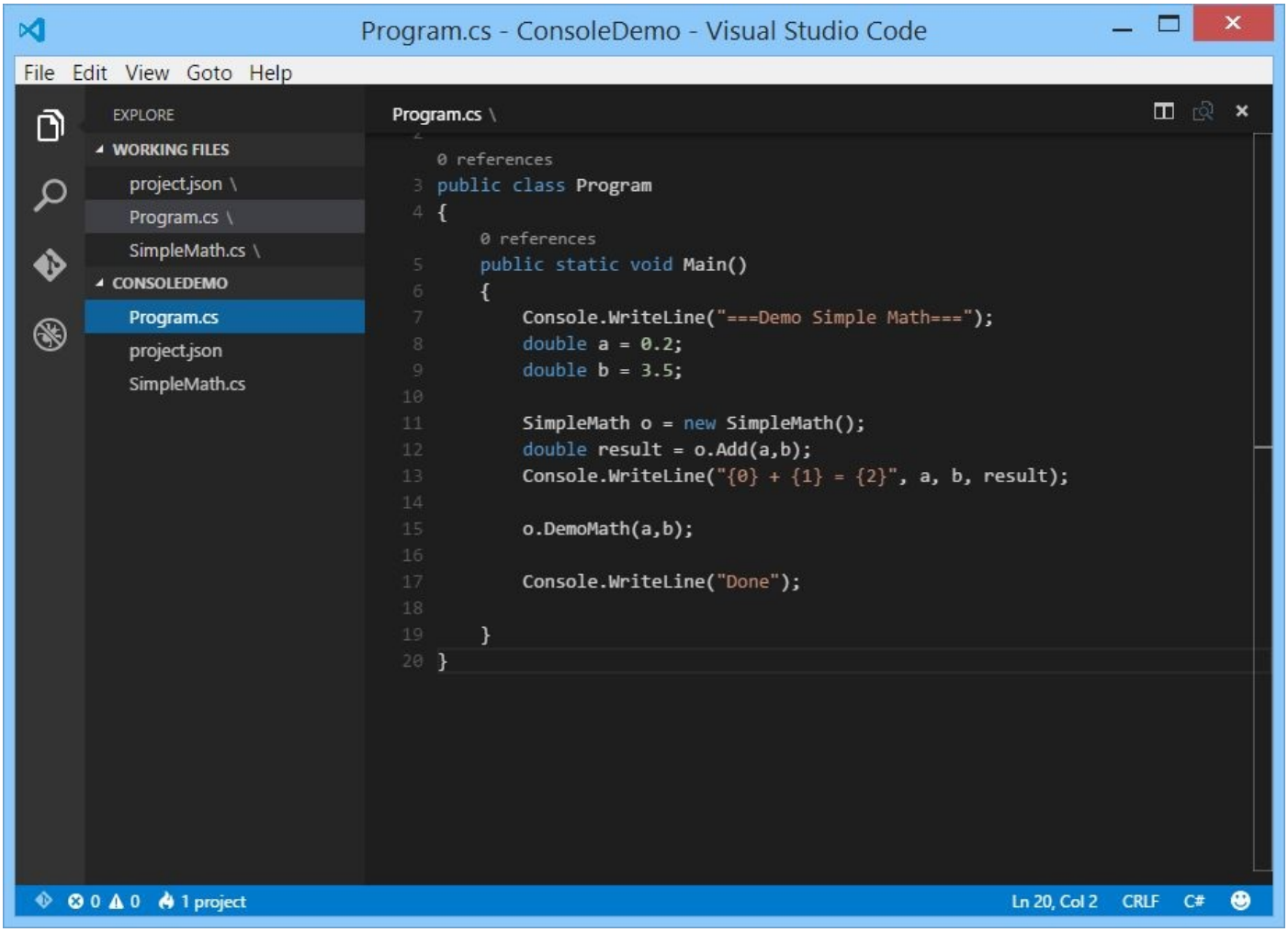
Now open a browser and then navigate to http://localhost:5004/ . You should see ASP.NET MVC app.



To stop kestrel, you can type exit.

4.4 Development Tools

The next step is to build program. To write a program, you can use any text editor. Microsoft also provides Visual Studio Code. It runs on Windows, Linux and Mac. You can download it on https://code.visualstudio.com/ .



5. ASP.NET 5 Development

In this chapter I’m going to explain how to work with ASP.NET 5 development.

5.1 Getting Started

In this chapter, we learn how to write a program for ASP.NET 5 from scratch. We use two scenarios, ASP.NET 5 Console and ASP.NET 5 MVC, for illustrating.

5.2 Building ASP.NET 5 Console Application

In this case, we build a simple program for ASP.NET 5 Console. We implement Math operation using Math class.

5.2.1 Creating A Project

To create a project, we can create a folder, called ConsoleDemo. In Linux/OS X, you can type this command on Terminal.

```
mkdir ConsoleDemo
```

Furthermore, we create a file, **project.json**. Type this script.

```
{
    "dependencies": {

    },
    "commands": {
        "ConsoleApp": "ConsoleDemo"
    },
    "frameworks": {
        "dnx451": { },
        "dnxcore50": {
            "dependencies": {
                "System.Console": "4.0.0-beta-*"
            }
        }
    }
}
```

This file is a project configuration which uses ASP.NET 5 DNX.

5.2.2 Writing Program

To create a program, we create a class, SimpleMath. Create a file, called **SimpleMath.cs**. Then, write this code.

```
using System;

public class SimpleMath
{
    public SimpleMath(){}
```

```
    public double Add(double a, double b)
    {
        return a + b;
    }
    public void DemoMath(double a, double b)
    {
        Console.WriteLine("cos({0}) = {1}", a, Math.Cos(a));
        Console.WriteLine("sin({0}) = {1}", b, Math.Sin(b));
        Console.WriteLine("exp({0}) = {1}", a, Math.Exp(a));
    }

}
```

The next step is to create a main program. Create a file, **Program.cs**, and write this code.

```
using System;

public class Program
{
    public static void Main()
    {
        Console.WriteLine("===Demo Simple Math===");
        double a = 0.2;
        double b = 3.5;

        SimpleMath o = new SimpleMath();
        double result = o.Add(a,b);
        Console.WriteLine("{0} + {1} = {2}", a, b, result);

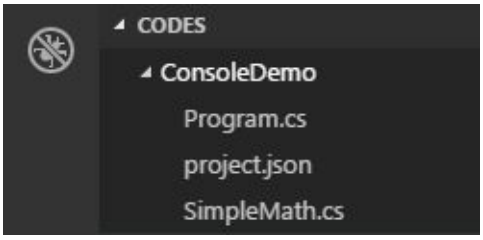
        o.DemoMath(a,b);

        Console.WriteLine("Done");

    }
}
```

Save all files.

The following is our project structure.

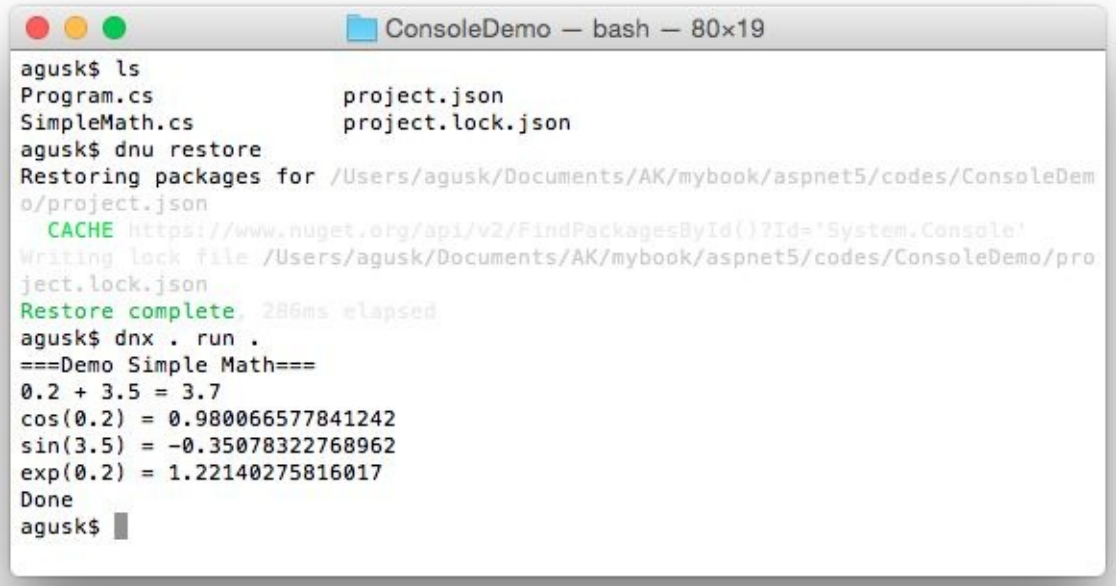


5.2.3 Testing

Now you can test the program. Firstly, we load all required libraries using dnu. Then, we execute the program using dnx.

```
dnu restore
dnx . run .
```

If success, you can see the program output.



5.3 Building ASP.NET 5 MVC Application

The second demo is to build ASP.NET 5 MVC application. We develop a simple entry form and utilize ASP.NET MVC session. The following is our scenario:

- User fills a form to create a new project
- User clicks Save button to save the data
- The data will be stored to a session stack
- Then, the data will be shown in another page

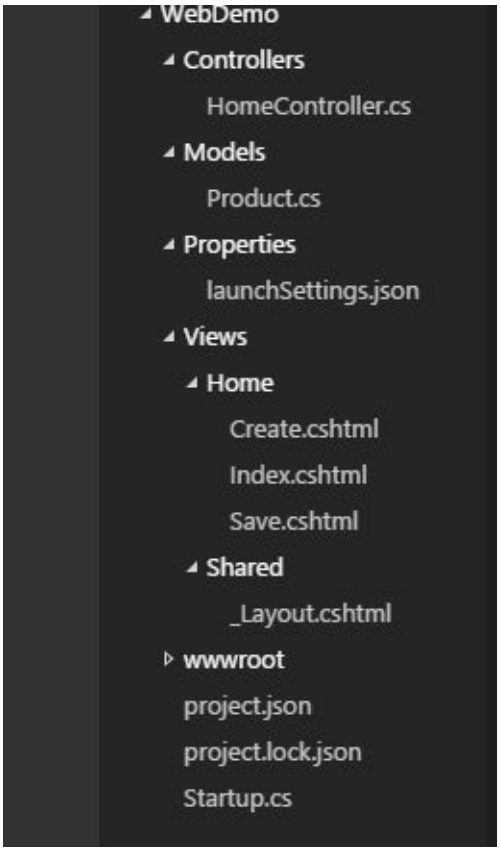
Let’s start to develop.

5.3.1 Creating a Project

Firstly, we create a project by creating a folder, called WebDemo. Then, you create the following folder:

- Controllers
- Models
- Properties
- Views
- wwwroot

You can see the project structure on the Figure below.



We need project a config file, called **project.json**, on the project root folder. It consists of required libraries. Please write this script for project.json file.

```
{
    "version": "1.0.0-*",
    "webroot": "wwwroot",
    "exclude": [
        "wwwroot"
    ],
    "packExclude": [
        "**.kproj",
        "**.user",
        "**.vspscc"
    ],
    "dependencies": {
        "Kestrel": "1.0.0-*",
        "Microsoft.AspNet.Diagnostics": "1.0.0-*",
        "Microsoft.AspNet.Mvc": "6.0.0-*",
        "Microsoft.AspNet.Server.IIS": "1.0.0-*",
        "Microsoft.AspNet.Server.WebListener": "1.0.0-*",
        "Microsoft.AspNet.Session": "1.0.0-*"
    },
    "commands": {
        "web": "Microsoft.AspNet.Hosting --server Microsoft.AspNet.Serve
        "kestrel": "Microsoft.AspNet.Hosting --server Kestrel --server.u
    },
    "frameworks": {
         "dnx451": { },
         "dnxcore50": { }
```

```
    }
}
```

App will start by executing **Startup.cs** file. Create this file and write this code.

```
using Microsoft.AspNet.Builder;
using Microsoft.Framework.DependencyInjection;

namespace WebDemo
{
    public class Startup
    {
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddMvc();
        }

        public void Configure(IApplicationBuilder app)
        {
            // handle error page
            app.UseErrorPage();
            // handle session
            app.UseInMemorySession();
            // aspnet routing
            app.UseMvc(routes =>
            {
                routes.MapRoute("default", "{controller=Home}/{action=In
            });

        }
    }
}
```

We also add **launchSettings.json** to configure how to execute the program. Store it on Properties folder. Write this script for **launchSettings.json** file.

```
{
  "profiles": {
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNET_ENV": "Development"
      }
    },
    "kestrel": {
      "commandName": "kestrel",
      "launchBrowser": true,
      "launchUrl": "http://localhost:5004"
    },
    "web": {
```

```
      "commandName": "web",
      "launchBrowser": true,
      "launchUrl": "http://localhost:5001"
    }
  }
}
```

5.3.2 Writing Program

In this section, we write our ASP.NET 5 MVC program.

5.3.2.1 Creating Model

A model holds a data. In our case, we create a module, called Product, which consists of four properties. Create a file, called **Product.cs**, and write this code.

```
using System.ComponentModel.DataAnnotations;

namespace WebDemo.Web.Models
{
    public class Product
    {
        [Required]
        [MinLength(4)]
        [Display( Name = "Name" )]
        public string Name { get; set; }
        [Display( Name = "Product Code" )]
        public string ProductCode { get; set; }
        [Display( Name = "Quantity" )]
        public int Quantity { get; set; }
        [Display( Name = "Is Discount" )]
        public bool IsDiscount { get; set; }
    }
}
```

Save this file into Models folder.

5.3.2.2 Creating Controller

Now we create MVC controller, called Home controller. Create a file, called **HomeController.cs**, and write this code.

```
using System;
using Microsoft.AspNet.Mvc;
using Microsoft.AspNet.Http;
using WebDemo.Web.Models;

namespace WebDemo.Web.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
        public IActionResult Create()
        {
            var product = new Product();
            return View(product);
        }
        [HttpPost]
        public IActionResult Create(Product obj)
        {
            // do database processing
            ///////

            // for testing, save it into a session
            Context.Session.SetString("name",obj.Name);
            Context.Session.SetString("code",obj.ProductCode);
            Context.Session.SetInt("quantity",obj.Quantity);
            Context.Session.SetString("discount",obj.IsDiscount.ToString

            return RedirectToAction("Save");
        }
        public IActionResult Save()
        {

            var product = new Product();
            product.Name = Context.Session.GetString("name");
            product.ProductCode = Context.Session.GetString("code");
            product.Quantity = (int)Context.Session.GetInt("quantity");
            product.IsDiscount = Convert.ToBoolean(Context.Session.GetSt

            return View(product);
        }


    }
}
```

5.3.2.3 Creating View

We create three pages:

- Index.cshtml
- Create.cshtml
- Save.cshtml

We also create a layout, called **_Shared.cshtml**. Write this script.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0
    <title>@ViewBag.Title - My ASP.NET Application</title>
    <link rel="stylesheet" href="//netdna.bootstrapcdn.com/bootstrap/3.0
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle=
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li><a href="/">Home</a></li>
                </ul>
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <footer>
            <p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>
        </footer>
    </div>
</body>
</html>
```

The following is a script for **Index.cshtml** file.

```
@{
    Layout = "/Views/Shared/_Layout.cshtml";
    ViewBag.Title = "Home Page";

}

<div class="jumbotron">
```

```
    <h1>ASP.NET MVC Demo</h1>
    <p class="lead"> Filling and submitting a form.</p>
    <p><a href="/Home/Create" class="btn btn-primary btn-large">Create &
</div>
```

The following is a script for **Create.cshtml** file.

```
@using WebDemo.Web.Models
@model Product
@{
    Layout = "/Views/Shared/_Layout.cshtml";
    ViewBag.Title = "Create Product";
}

<div class="jumbotron">
    <h1>ASP.NET</h1>
    <p class="lead">Create a new product. Fill this form. Then, click Sa

</div>
@using ( Html.BeginForm() )
    {
        <fieldset>
                <legend>Create a new Product</legend>

                <div class="editor-label">
                        @Html.LabelFor(model => model.Name)
                </div>
                <div class="editor-field">
                        @Html.TextBoxFor(model => model.Name)
                </div>
                <br/>

                <div class="editor-label">
                        @Html.LabelFor(model => model.ProductCode)
                </div>
                <div class="editor-field">
                        @Html.TextBoxFor( model => model.ProductCode )
                </div>
                <br/>
            <div class="editor-label">
                        @Html.LabelFor(model => model.Quantity)
                </div>
                <div class="editor-field">
                        @Html.TextBoxFor( model => model.Quantity )
                </div>
                <br/>

                <div class="editor-label">
                        <div style="float: left;">
                                @Html.CheckBoxFor(model => model.IsDisco
                        </div>
                        @Html.LabelFor(model => model.IsDiscount)
```

```
                </div>
                <br/>

                <input type="submit" value="Save" />

        </fieldset>
    }
<p><br></p>
```

The following is a script for **Save.cshtml** file.

```
@model WebDemo.Web.Models.Product
@{
    Layout = "/Views/Shared/_Layout.cshtml";
    ViewBag.Title = "Reading Session";
}

<div class="jumbotron">
    <h1>ASP.NET</h1>
    <p class="lead">Reading data from ASP.NET MVC session.</p>
    <p><a href="/Home" class="btn btn-primary btn-large">Back to home &r
</div>
<div>
    <p>Name: @(Model != null ? Model.Name : "")</p>
    <p>Product Code: @(Model != null ? Model.ProductCode : "")</p>
    <p>Quantity: @(Model != null ? Model.Quantity : 0)</p>
    <p>Is discount: @(Model != null ? Model.IsDiscount : false)</p>
</div>
<p><br></p>
```

Save all files and store them into Views folder. See project structure.

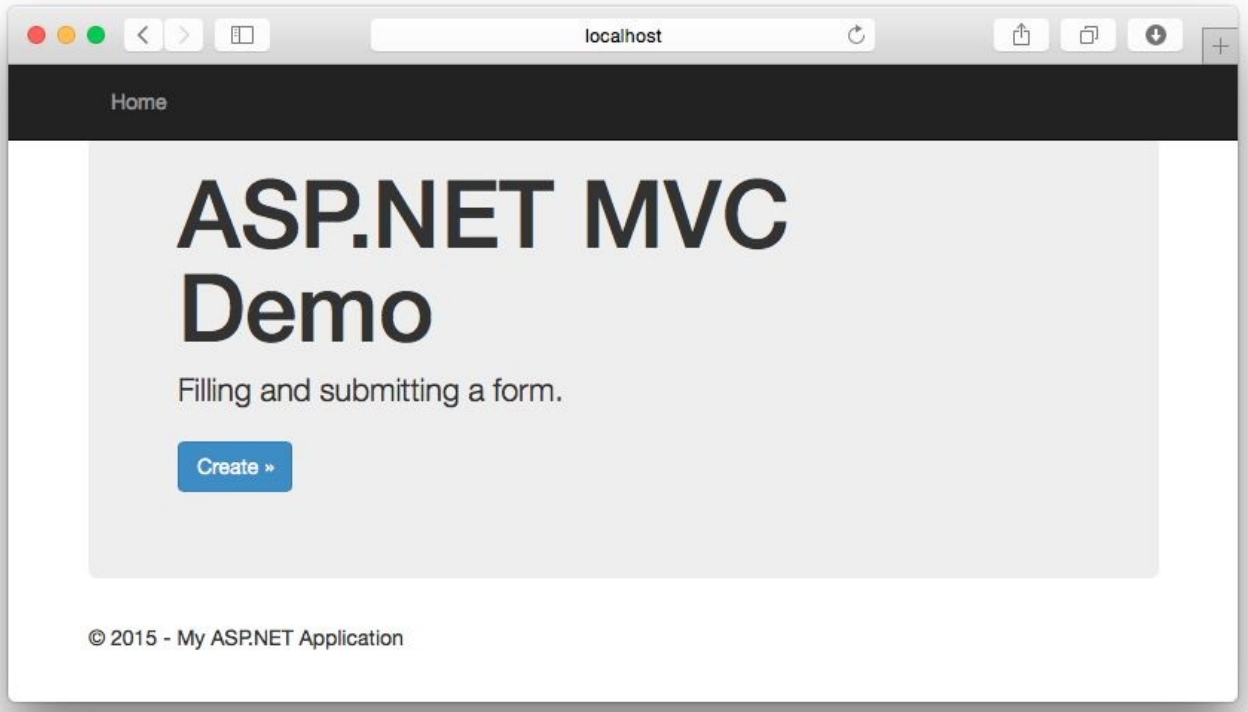
5.3.3 Testing

Now you are ready to execute. Type these command to load required libraries and execute the program.

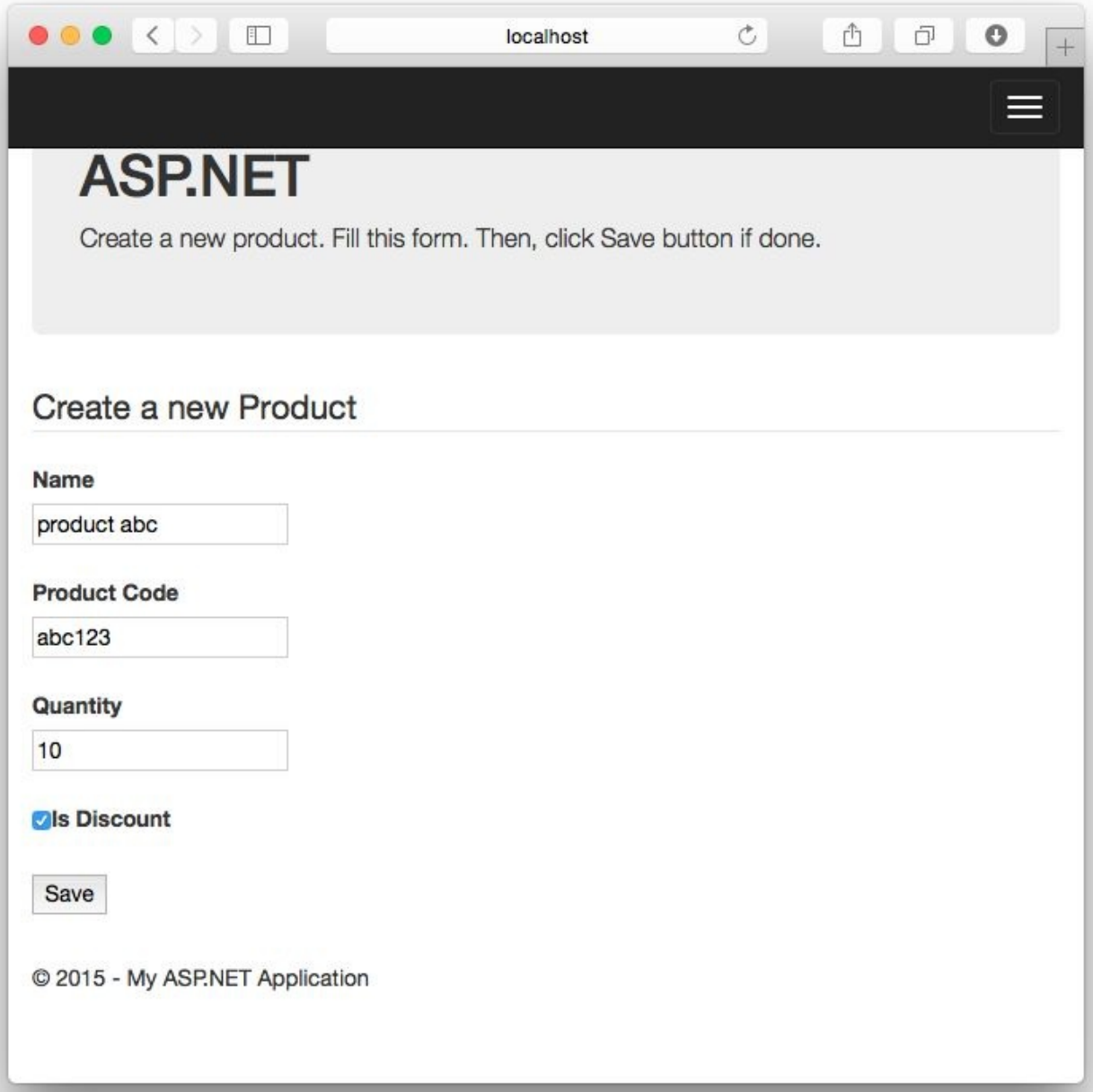
```
dnu restore
dnx . kestrel
```



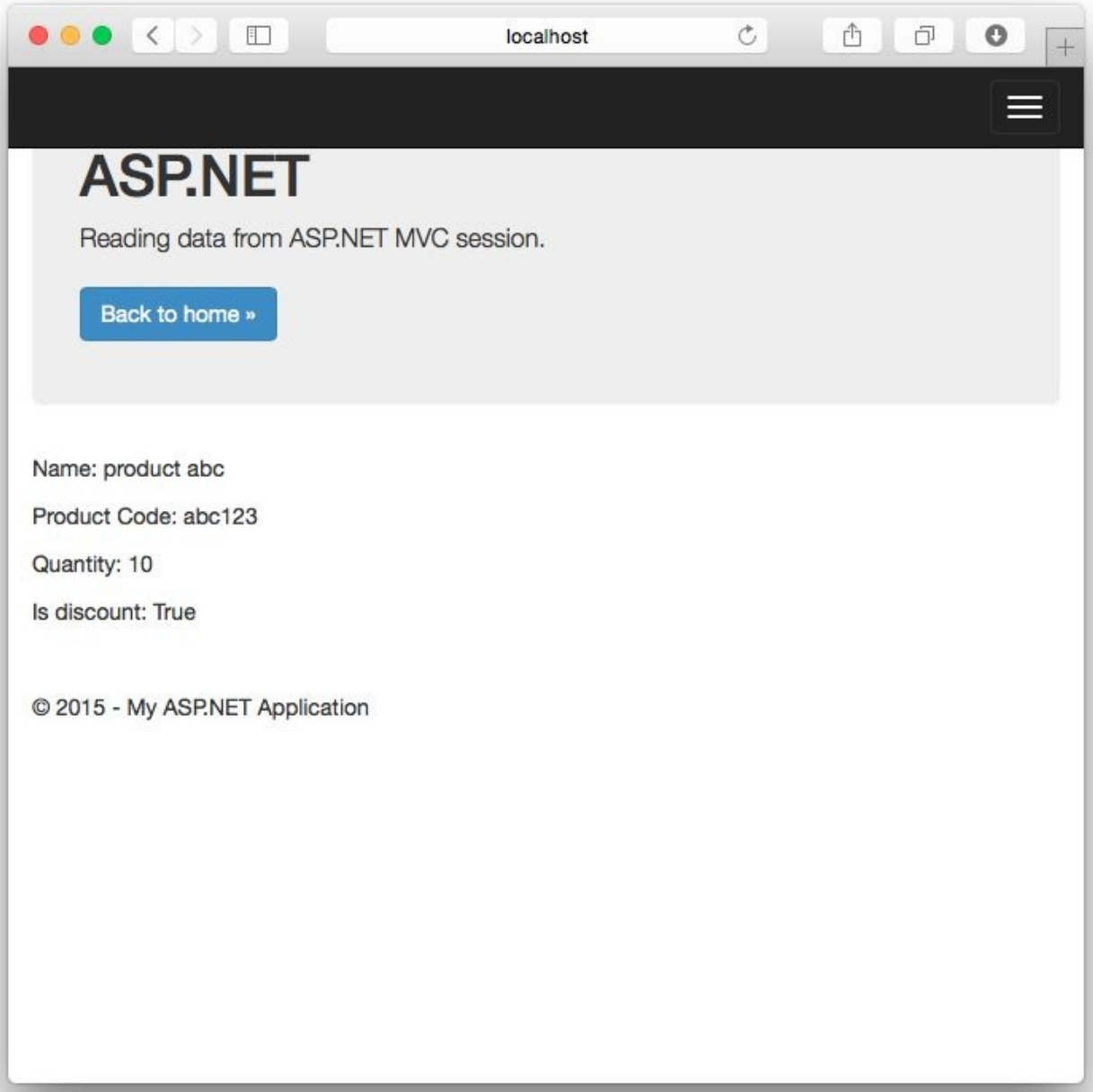
Open a browser and navigate to http://localhost:5004 . Click **Create** button.



Fill all fields. If done, click **Save** button.



Our filled form will be shown in the next page. It uses ASP.NET session to keep our filled data.


localhost
ASP.NET
Reading data from ASP.NET MVC session.
Back to home »
Name: product abc
Product Code: abc123
Quantity: 10
Is discount: True
© 2015 - My ASP.NET Application

6. ASP.NET 5 RESTful and AngularJS

In this chapter I’m going to explain how to work with ASP.NET 5 RESTful and AngularJS

6.1 Getting Started

In this section, we develop a simple app to work with ASP.NET RESTful and AngularJS.

AngularJS is an open-source JavaScript framework, maintained by Google, that assists with running single-page applications. Further information about AngularJS, you can visit official website on http://angularjs.org/ .

To integrate your application with AngularJS, you can download AngularJS on http://code.angularjs.org/. Choose AngularJS version you want to use. To use AngularJS on your web application, you just put it on your web page:

```
<script src="lib/angular.min.js"></script>
```

This means AngularJS file will be downloaded from your local. You also can put AngularJS as follows.

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.15/angu
```

The following is our scenario:

- User click a button
- Program will request data from server via RESTful

Let’s start to develop.

6.2 Creating Project

Firstly, we create a project by creating a folder, called RESTfuldemo. After that, you can create the following folder:

- Controllers
- Database
- Models
- Properties
- Views
- wwwroot

The following is project structure.



We need project config file, called **project.json**, on the project root folder. It consists of required libraries. Please write this script for **project.json** file.

```
{
```

```
    "version": "1.0.0-*",
    "webroot": "wwwroot",
    "exclude": [
        "wwwroot"
    ],
    "packExclude": [
        "**.kproj",
        "**.user",
        "**.vspscc"
    ],
    "dependencies": {
        "Kestrel": "1.0.0-*",
        "Microsoft.AspNet.Diagnostics": "1.0.0-*",
        "Microsoft.AspNet.Mvc": "6.0.0-*",
        "Microsoft.AspNet.Server.IIS": "1.0.0-*",
        "Microsoft.AspNet.Server.WebListener": "1.0.0-*",
        "Microsoft.AspNet.StaticFiles": "1.0.0-*"
    },
    "commands": {
        "web": "Microsoft.AspNet.Hosting --server Microsoft.AspNet.Serve
        "kestrel": "Microsoft.AspNet.Hosting --server Kestrel --server.u
    },
    "frameworks": {
         "dnx451": { },
         "dnxcore50": { }
    }
}
```

We also add **launchSettings.json** to configure how to execute the program. Store it on Properties folder. Write this script for **launchSettings.json** file.

```
{
  "profiles": {
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNET_ENV": "Development"
      }
    },
    "kestrel": {
      "commandName": "kestrel",
      "launchBrowser": true,
      "launchUrl": "http://localhost:5004"
    },
    "web": {
      "commandName": "web",
      "launchBrowser": true,
      "launchUrl": "http://localhost:5001"
    }
  }
}
```

App will start by executing **Startup.cs** file. Create this file and write this code.

```
using Microsoft.AspNet.Builder;
using Microsoft.Framework.DependencyInjection;

namespace WebDemo
{
    public class Startup
    {
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddMvc();
        }

        public void Configure(IApplicationBuilder app)
        {
            // handle error page
            app.UseErrorPage();
            // handle static file requests
            app.UseStaticFiles();
            // aspnet routing
            app.UseMvc(routes =>
            {
                routes.MapRoute("default", "{controller=Home}/{action=In
            });

        }
    }
}
```

6.3 Writing Program

In this section, we write our ASP.NET 5 MVC program.

6.3.1 Creating Model

Firstly, we create a model, called Product. The following is code implementation.

```
using System.ComponentModel.DataAnnotations;

namespace WebDemo.Web.Models
{
    public class Product
    {
        [Required]
        [MinLength(4)]
        [Display( Name = "Name" )]
        public string Name { get; set; }
        [Display( Name = "Product Code" )]
        public string ProductCode { get; set; }
        [Display( Name = "Quantity" )]
        public int Quantity { get; set; }
        [Display( Name = "Price" )]
        public double Price { get; set; }
    }
}
```

Save this code into a file, called **Product.cs** and store it into Models folder.

6.3.2 Dummy Database

In this case, we don’t access database directly so we create a dummy database which generates product data. Create a file, called **DumpDatabase.cs** and write this code.

```
using System;
using System.Collections.Generic;

using WebDemo.Web.Models;
namespace WebDemo.Web.Database
{
    public class DumpDatabase
    {
        private DumpDatabase(){}
                public static List<Product> GetAllProducts()
```

```
                {
                        // generate data
                        List<Product> list = new List<Product>();
                        for(int i=0;i<10;i++){
                                var item = new Product();
                                item.Name = String.Format("Product {0}",
                                item.ProductCode = String.Format("K0{0}"
                                item.Quantity = 4 + i;
                                item.Price = 1.43 * (i+1);

                                list.Add(item);

                        }

                        return list;
                }

    }
}
```

Save this file into Database folder.

6.3.3 Creating Controller

Furthermore, we build a controller, called Home by creating a file, called **HomeController.cs**. Write this code.

```
using System;
using System.Collections.Generic;
using Microsoft.AspNet.Mvc;

using WebDemo.Web.Models;
using WebDemo.Web.Database;
namespace WebDemo.Web.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }

        [HttpPost]
        public JsonResult GetAllProducts()
        {
            List<Product> list = DumpDatabase.GetAllProducts();

            return Json(list);
```

```
        }


    }
}
```

Save this file into Controllers folder.

6.3.4 Creating View

We create a page, Index.cshtml. We also create a layout, called **_Shared.cshtml**. Write this script.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0
    <title>@ViewBag.Title - My ASP.NET Application</title>
    <link rel="stylesheet" href="//netdna.bootstrapcdn.com/bootstrap/3.0
    <link href="~/css/simple.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle=
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li><a href="/">Home</a></li>
                </ul>
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <footer>
            <p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>
        </footer>
    </div>
</body>
</html>
```

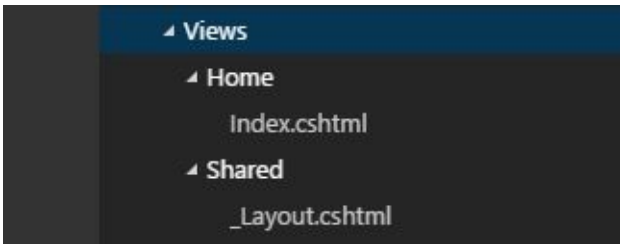
The following is a script for **Index.cshtml** file.

```
@{
    Layout = "/Views/Shared/_Layout.cshtml";
    ViewBag.Title = "Home Page";

}
<script src="~/js/lib/angular.min.js"></script>
<script src="~/js/services/productservice.js"></script>
<div class="jumbotron">
    <h1>ASP.NET 5 MVC, ASP.NET RESTful, and AngularJS</h1>
    <p class="lead"> This is demo to illustrate how to use ASP.NET 5 MVC
</div>
<h2>Product Application</h2>
<br />
<div ng-app="ProductService" ng-controller="MyControlller" >
    <p>
        <button class="btn btn-primary btn-large" ng-click="getAllData()
    </p>
    <table>
        <thead>
            <tr>
                <th>Product Name</th>
                <th>Code</th>
                <th>Quantity</th>
                <th>Price</th>
                <th>Action</th>
            </tr>
        </thead>
        <tbody ng-repeat="product in products">
            <tr ng-class="{even: $even, odd: $odd}">
                <td>{{ product.Name }}</td>
                <td>{{ product.ProductCode}}</td>
                <td>{{ product.Quantity}}</td>
                <td>{{ product.Price | number:2}}</td>
                <td>
                    <a href="javascript:void(0)" ng-click="select(produc
                </td>
            </tr>
        </tbody>
    </table>
    <p>Selected: {{ selected }}</p>
</div>
<p><br></p>
```

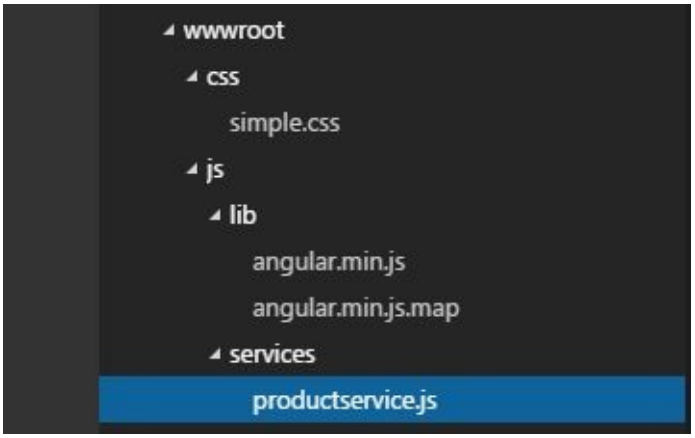
This file loads AngularJS files. We write them on next section.

Save all files. The following is our View structure.



6.3.5 AngularJS Program

Firstly, download AngularJS on http://code.angularjs.org/. Put those files into /js/lib folder.



The next step is to write AngularJS program. Create a file, called **productservice.js**, and write this script.

```
var myApp = angular.module('ProductService', []);

myApp.service('DataService', function ($http) {

    this.read = function (callback) {
        $http({
            method: 'POST',
            url: 'Home/GetAllProducts/',
            headers: {
                'Content-type': 'application/json'
            }
        }).success(function (resp) {
            callback(resp);
        }).error(function () {
            callback(undefined);
        });

    };

});


myApp.controller('MyControlller', function ($scope, DataService) {
```

```
    $scope.products = [];
    $scope.selected = {};

    $scope.getAllData = function () {
        DataService.read(function (data) {
            $scope.products = data;
        });

    };
    $scope.select = function (selected) {
        $scope.selected = selected;
    };
});
```

The following is a script for **simple.css** file.

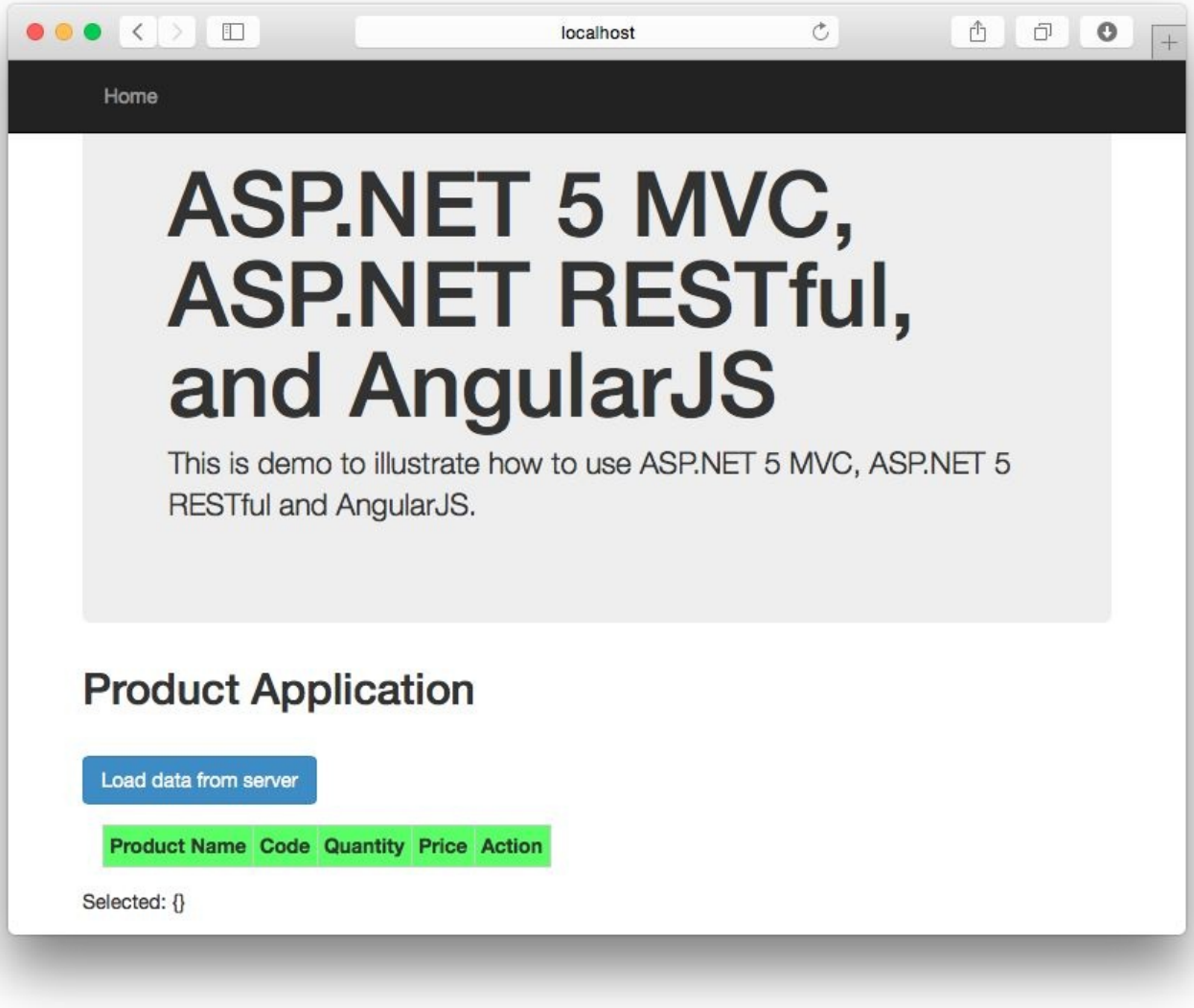
```
table { margin: 1em; border-collapse: collapse; }
td, th { padding: .3em; border: 1px #ccc solid; }
thead { background: #70ff45; }
.even{
    background-color: #beffca;
}
.odd{
    background-color: #f0fff6;
}
```

6.4 Testing

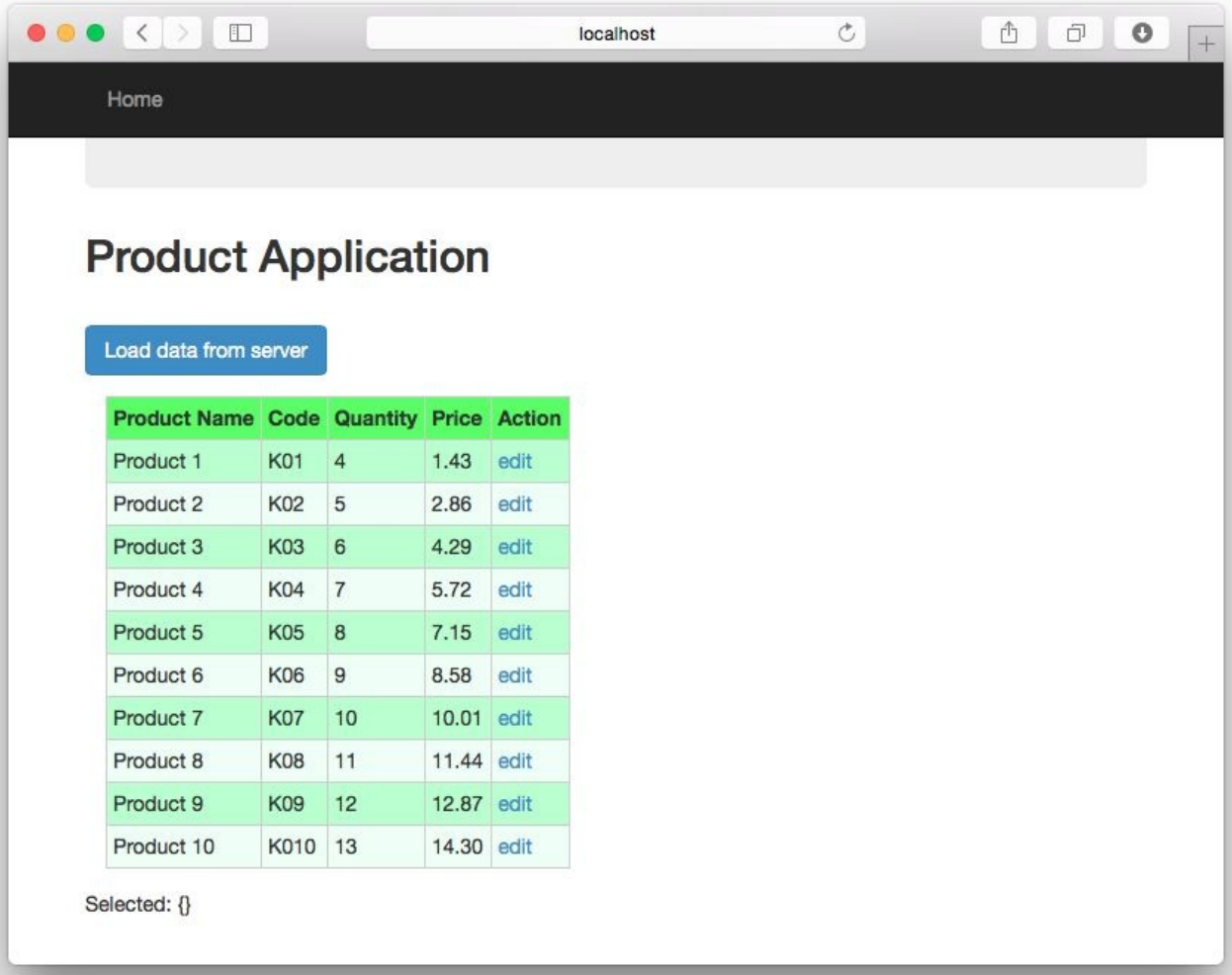
Now you are ready to execute. Type these command to load required libraries and execute the program.

```
dnu restore
dnx . kestrel
```

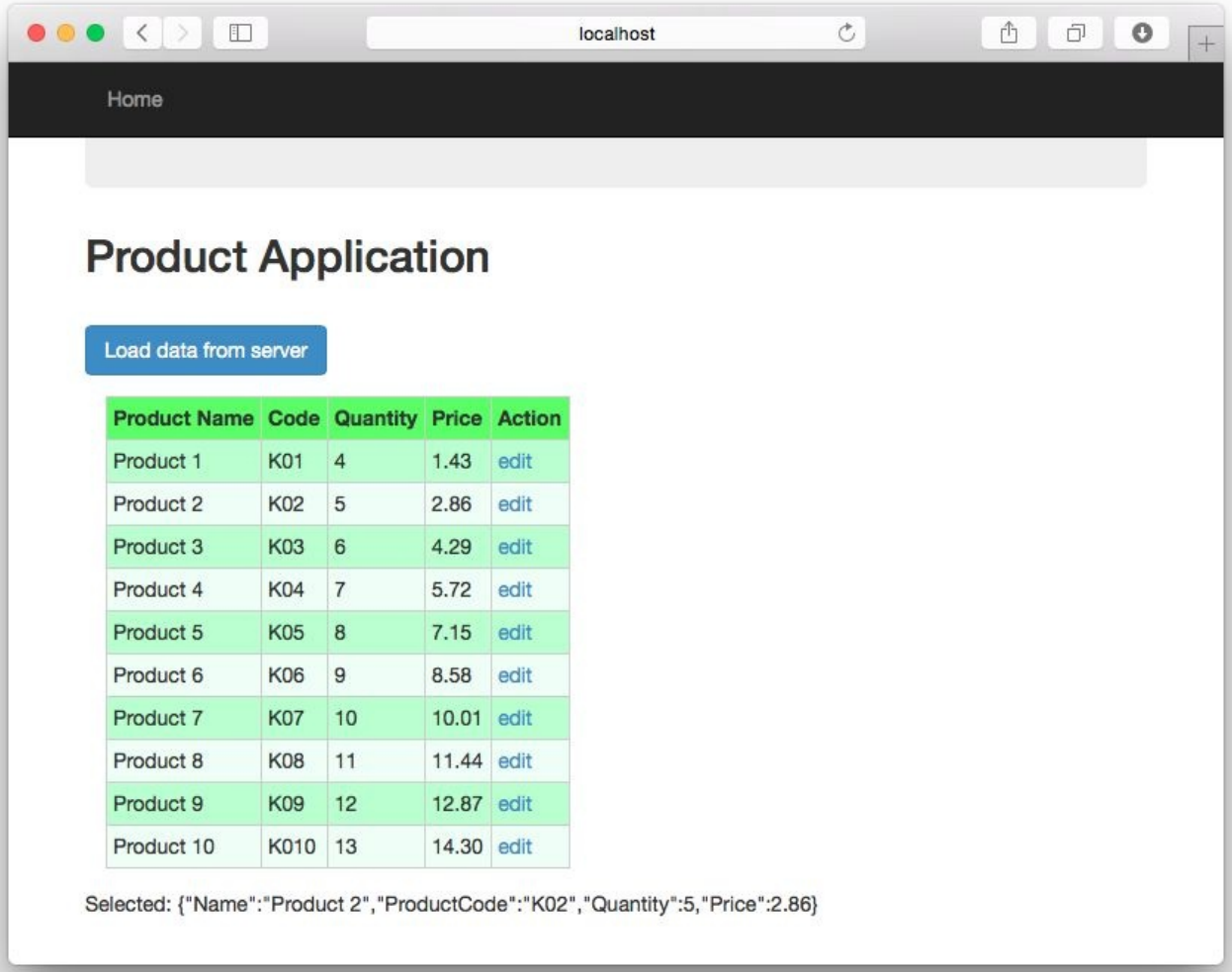
Open a browser and navigate to http://localhost:5004 .



Click Load data from server button. It should show data which come from a server.

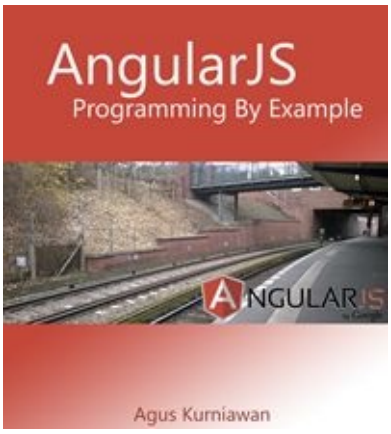


Try to click **edit** menu on table row. Then, selected item will show on the bottom.


localhost
Home
Product Application
Load data from server
Product Name Code Quantity Price Action
Product 1 K01 4 1.43 edit
Product 2 K02 5 2.86 edit
Product 3 K03 6 4.29 edit
Product 4 K04 7 5.72 edit
Product 5 K05 8 7.15 edit
Product 6 K06 9 8.58 edit
Product 7 K07 10 10.01 edit
Product 8 K08 11 11.44 edit
Product 9 K09 12 12.87 edit
Product 10 K010 13 14.30 edit
Selected: {"Name":"Product 2","ProductCode":"K02","Quantity":5,"Price":2.86}

6.5 What’s The Next?

You can do more practices for AngularJS by reading several tutorial in Internet or books. I also wrote a book about AngularJS Programming by Example. This book helps you to learn about AngularJS programming. Please visit my blog, http://blog.aguskurniawan.net/post/AngularJS.aspx .



Source Code

Source code can be downloaded on
http://www.aguskurniawan.net/book/aspnet5_06062015ZFR.zip.zip.

Contact

If you have question related to this book, please contact me at aguskur@hotmail.com . My blog: http://blog.aguskurniawan.net.